ASK French

A French Natural Language Syntax

Remy Sanouillet

Computer Science Department
California Institute of Technology

5164:TR:84

ASK FRENCH

A FRENCH NATURAL LANGUAGE SYNTAX

by Remy Sanouillet

In Partial Fulfillement of the Requirements

for the Degree of

Master of Science

California Institute of Technology
Pasadena, California
October 1984

5164:TR:84

ASK French, a French Natural Language Syntax

Extending information management systems from the standard English user interface to a multilingual interface, thus allowing users of different countries to access a same shared data base in their mother tongue, required the creation of a foreign grammar for implementation purposes. This has led to the generation of ASK French, a counterpart to the english information processing system: ASK.

A typical session with ASK is basically composed of a series of querries from the user. Each of these querries goes through two major phases, the SYNTAX phase, then the SEMANTIC phase and finally results in an appropriate reply from the system. During the SYNTAX phase, the system tries to parse the input by comparing the sentence's structure to its set of grammar rules searching for a match. Each grammar rule is linked to a semantic procedure, (a Pascal program), which is executed during the SEMANTIC phase.

An example of a simple querry follows, where the user's input is preceded by a ")" symbol, the reply from the system is typed immediatly below. On the left is a ASK-French querry, on the right is its ASK-English equivalent.

```
)Quels vins rouges sont des Bourgognes?
 Gevrey-Chambertin
 Nuits-Saint-Georges
 Pommard
 Volnay
 Meursault
 Pouilly-Fuisse
 Clos-Vougeot
 Macon
```

```
)What red wines are Burgundies?
 Gevrey-Chambertin
 Nuits-Saint-Georges
 Pommard
 Volnay
 Meursault
 Pouilly-Fuisse
 Clos-Vougeot
 Macon
```

The subject of this paper is to discuss the main grammatical differences between ASK-English and ASK-French computer designed syntax, the requirements it puts on the semantics and the techniques used to generate appropriate responses in the language of the user. Of course the complexity of such a task does not reach the linguistic levels of machine translation in terms of the rather simple and rigid structures of the the data base, thus avoiding subtle aspects of human language understanding, since only the intent of the querry is understood. The underlying semantic interpretation of the request is designed to be language independent so as to enable us to use the same semantic procedures for both languages, and facilitate the addition of other languages.

In fact, the major part of ASK's semantics consist of data structure manupilation routines. The data structures are stored using a semantic net with specific entities at the nodes and relations as edges. Hence ASK has a very deterministic approach to data base storage and handling mechanisms, which makes it especially well adapted to encompass traditional business data base structures, e.g. relational, hierarchical, etc... But since for these reasons, it would be detrimental to the system in terms of clarity, response times and specificity, to incorporate "fuzziness" or probabilistic domains in the data base, ASK can only understand a small subset of natural language. For example, ASK-French cannot handle sentences such as:

>Le resultat du jet d'un de est compris entre un et six.
(The outcome of the roll of a die is between one and six.)

It will be able to parse it, and syntactically analyze it, but when the appropriate semantic routine will try to assign a specific numeric value to the entity "outcome of a die", it will fail by not being able to assimilate "between one and six" as a definite number or being able to understand that an equiprobable set of numbers is a valid result.

The limitations of ASK-French compared to natural French can appear in many other similar manners. The main goal of this grammar not being to understand the whole of French as a natural language, but to give to the user a very flexible tool, the basics of which he has already mastered since a young age; a tool that will allow him to work effectively on an information ensemble with a minimum amount of initiation and constraints.

Main syntactical differences between ASK-French and ASK-English

French and English have distinct roots, French is of Latin origin and English is of Germanic origin. But the two languages have similarities due to three main reasons.

First, they both have the same proto-language, namely Indo-european.

Second, England was governed by an Anglo-Normand court for the whole period between the battle of Hastings (1066) and Magna Carta (1215). During that period, "the French language and French usages were introduced and spread downwards through the free classes." [Barlow] This influence can be found in the Royal Arms of England, where Richard the Lion-Heart's motto "Dieu et mon Droit" is inscribed.

Third, the actual trend is reversed. "The number of English loan words in modern French is rising so rapidly that it is becoming a cause for alarm in some quarters (although it is hard to see why such a natural phenomenon as lexical borrowing should be considered alarming)." [Langacker]

Nouns

French nouns as opposed to most English ones have genders. The gender of the noun effects the use of the article that precedes it, the spelling of the adjectives and past participles that reffer to it, and the use of pronouns that replace it.

The problem that arises is: How is the user to specify the gender of the new words he/she is adding to the system.

>Cree un individu nomme Jean
(Create un individual named John)

The preceding sentence is the standard way one can add new vocabulary items to ASK-French. This sentence doesn't contain enough syntactical information for ASK-French to determine the gender of the new word "Jean".

One solution would be to transfer the burden onto the user, by requiring that he specify the gender of every word he/she creates, as in the following example:

>Cree des individus masculins nommes Jean, Marc et Henri
(Create masculine individuals named John, Mark and Henry)

This solution has been rejected for two main reasons. First, the user is now forced to define only words of the same gender at a time. Hence he/she could not define John, Ted and Alice in a single sentence. It is also a main policy in ASK to be as user-friendly as possible, and this solution would unnecessarily constrain the user.

The solution adopted in ASK-French begins by giving any new word an indefinite gender (not to be confused with the neutral gender found in other languages). Then the first time that word is used in a sentence that gives a clue to its gender, the system automaticaly adds that new gender to it, so that from there on, the computer can generate grammatically correct sentences by making articles, etc. to agree with the noun. Let us go through a typical session and see how the gender is assigned.

```
)Cree un individu nomme Anne
 Le nouvel individu  Anne  a ete ajoute.

)Cree une categorie nommee femme
 La nouvelle categorie  femme  a ete ajoute.
(Create a class named female
 The new class female has been added
 Note: at this stage the entry for female in the lexicon
 indicates it is of indefinite gender)

)Anne est une femme.
 Anne a ete ajoute a la categorie femme.
(Ann is a female.
 Ann has been added to the class female.
 Note: now the entry for female in the lexicon has been
 changed to include the feature for feminine gender)
```

Articles

French articles don't always carry the gender information. If definite articles such as "le" (masculine) and "la" (feminine) are clear indicators of the word's gender, "les" (plural) and "l'" (contracted article used when the noun begins with a vowel or a mute h in graphemics) are of no help. In this last case, there is little else to be done than to ignore the article.

The case is similar for indefinite articles and partitive articles: "un" and "du" are masculine, "une" and "de la" are feminine, "des" is plural and does not indicate gender, and "de l'" is contracted.

The syntactic rules for articles that carry gender are shown below. For explanations on how to interpret these rule, see the appendix.

```
RULE "le"
(article_defini:+mas) => "le"
LEX 0

RULE "la"
(article_defini:+fem) => "la"
LEX 0

RULE "un"
(article_indefini:+mas) => "un"
LEX 0

RULE "une"
(article_indefini:+fem) => "une"
LEX 0
```

Contraction, redundancy and euphonics

Contraction occurs for almost any word that precedes a noun beginning with a vowel or a mute h in graphemics. Prepositions, quantifiers, pronouns, articles all have contracted states. Handling of contraction would require doubling all grammar rules using those words; one rule for the uncontracted case with a blank between the word and the noun, the other rule when the contraction requires no spacing. To prevent such an inefficient amount of rules to be generated, ASK-French parses contractable words and separators into a new entity which are rules covering their further usage.

The French language contains many features that differ from English. One of the more distinctive is the negation form. A negation of a French sentence is formed by taking the positive form and putting a "ne" before the verb and a "pas" after. Somewhat similarly, a large portion of the prepositions used in everyday language have become parts of expressions, such as "de" in "combien de" (how many). ASK-French parses the "ne" in negations and these prepositions in a very similar manner as for articles with no gender information. They are included in the parsing graph and are carried along to help in the morphological construction of the response, but their very small semantical meaning is ignored by the deep structure analysis process.

The case is identical for euphonic characters, i.e. characters that are introduced in between two words to ease the pronunciation, such as the "-t-" in the interrogation structure "y a-t-il" (is there).

Genders, as an extra source of semantical information

The gender can also be used as a disambiguating help to find pronoun referents. Consider the sentence: "When the boy kissed the girl, she slapped him." There is little doubt because of our knowledge of the world, about who got slapped. But ASK doesn't know, unless it is told so, that "girl" is feminine. There is no syntactical clue in the sentence, that could tell that "she" reffers to "the girl". In French, the article would give the identity of the slapper, the translated sentence being "Quand le garcon embrassa la fille, elle le gifla."

Plurals

The general rule for plurals is the same in both languages. A plural is formed by tagging an "s" at the end of the noun. The exceptions are a different matter altogether.

There are three major exceptions in English which are handled in ASK-English:
- Words ending in o, s, x, z, sh or ch usually form their plural in es.
- Words ending in a y preceded by a consonant form their plural in ies.
- Words ending in fe and some words ending in f form their plural in ves.

The first case is easily taken care of by tolerating both forms of plural. For example "potatos" and "potatoes" will be equally understood, even though the former is not grammatical. This is still in the spirit of making, the system as user-friendly and as grammatically tolerant as we can without impairing the clarity of the language. This relaxation of grammatical rules is a key factor in making the system habitable. This general leniancy of course does not apply to the computer's responses.

The second and third case can not be handled in the same manner since the plural form is a modified copy of the root word, and not only a suffix that is added on. To take care of this problem, the system looks for special endings like "v" and "fe" when words are created by the user. It then stores in its lexicon, not the whole word like it would for a normal input, but the part of the word that is invariable and gives it the part of speech noun stem. Then when that particular noun stem appears in a sentence, it looks at the suffix and determines the case.

The situation is similar in French except that some words are invariable, namely those ending in s, x or z. The second most common plural ending after s is x. It is added to words ending in au, eu and some in ou. Words ending in al and ail are the ones requiring noun stem handling since their plural is aux. Other exceptional cases have to be handled directly by the user using the definitional capabilities of the system. There are so many exceptional situations in French that to include them all would seriously impair the response time of the system.

Furthermore, users have a common tendancy of using personnal jargon, which often have strange plurals. This can obviously not be accounted for by the basic grammar. The user specifies these plural in the following manner. Take the example of the plural of French compound words which fill the nightmares of young French students. The rules for forming those plurals are marred with exceptions and exceptions to the exceptions. For example, if the user created a class called "grand-pere" (grandfather), ASK-French would not be able to handle its plural "grands-peres" because of the "s" introduced in the middle. The user only has to type:

```
>definition: grands-peres : grand-peres
```

for the system to be able to handle that special case. Similarly, to handle the plural of "oeil" (eye) which is "yeux" the user would type:

```
>definition: yeux : oeils
```

Another use of definitions is found in conjugating irregular verbs. ASK understands the most common conjugations of verbs. But to add all conjugations of verbs in the grammar would increase its size to unefficient proportions, for the same reasons all plurals can't be taken care of. There are more than a hundred types of irregular verbs.

Hence, the definitional capabilities of ASK are a very powerful way of handling exceptional grammatical cases. This, of course, is not their only use.

Feminine Gender

The feminine of nouns or adjectives is generally formed in French by adding an "e" at the end. There are few common exceptions, so that they can be handled by definitions. The feminine suffix has precedence over the plural suffix, so that a word ending in "es" is determined to be feminin plural, while one ending in "se" is not. Sometimes a root word with an "e" suffix is not intended to be the feminine form of that word. For example, if a user created two individuals called Pascal and Pascale, any reference in a sentence to Pascale would be ambiguous. The first meaning being Pascale, the second being the feminin version of Pascal. To avoid that, the system checks newly created words for an "e" suffix. If there is one it checks in its lexicon to see if a version of the word without the suffix already exists. If so, it sets a special feature on that word to disallow any feminine parsing of it. If the created word does not have the "e" suffix, it looks for an already existing word with the suffix on and adapts the new word accordingly.

Special characters and "accents"

Non ASCII characters are quite common among the european languages, however most modern computers offer an extended ASCII character set, and modified keyboards to enable the input and output of these characters. So that given the necessary hardware, ASK is fully prepared to handle most of the european languages.

Word order

Word order usually has repercussions very deep into the system, often downto the semantic level. semantic routines are shared by all languages. When a given semantic concept is expressed by phrases with different word order for two languages, for example:

red apple

pomme rouge,

this word order difference can be handled at the very beginning of the associated semantic procedure.

The system looks at an internal flag that specifies the language the users is using, and unifies the semantical structure of the phrase.

Adjectives in French have various positions in reference to the noun they modify. When they are attributes, the position is identical to the one in English, which is after the verb: "La pomme est rouge." (The apple is red.). If they are epithets though, they can be found on either side of the noun, according to complex grammatical rules. The internal representation of an adjective in ASK is identical to the one for a noun. This is no problem in English since the epithet is always placed before the noun (the red apple). In French, the only way to find out where the adjective is, is to analyse the underlying semantical structure of the sentence and checking the result against the data base.

A case where the English structure is more complex than French is the possessive. There is only one possessive structure in French, namely "possessed" of "possessor"; e.g. "le nom de la fille" (the name of the girl), even though structures with the preposition "pour" are very close semantically to a possessive, e.g. "Une bonne annee pour les vins." (A good year for wine) and "Une bonne annee des vins" (A wine's good year). English also has the contracted structure "possessor"'s "possessed"; e.g. the girl's name.

Cardinal and ordinal numbers

The structure of ordinal numbers in French is somewhat related to the one in English, since they both use the arabic representation for cardinal numbers. One minor difference is that for large numbers, French omits the "one" in front, e.g. one thousand is "mille" but two thousand is "deux mille". This kind of differences is solved by changing the precedence features in the grammar rules, as is shown below.

French rules

```
RULE hundred thousand
⟨number:+lit+big+lit3⟩ => ⟨number:+lit+big-lit2-lit3⟩ " "
                          ⟨number:+lit+big-lit2⟩
SYN 2 num_mult

RULE thousand hundred
⟨number:+lit+big+lit3⟩ => ⟨number:+lit+big-lit2-lit3⟩ " "
                          ⟨number:+lit⟩
SYN 2 num_add
```

English rules

```
RULE one thousand one hundred
⟨whole_number:+big+lit2+lit3⟩ => ⟨whole_number:+big+lit2-lit3⟩ " "
                                 ⟨whole_number:+big+lit2⟩
SYN 2 wh_num_add

RULE hundred thousand
⟨whole_number:+big+lit3⟩ => ⟨whole_number:+big-lit2-lit3⟩ " "
                            ⟨whole_number:+big-lit2⟩
SYN 2 wh_num_mult
```

Another difference, but which has to be solved semantically, is the following. Consider the following sentence: "The cars have twenty and one painted on the door". The semantical interpretation of this can be pictured as a group of cars with the numbers 20 and 1 on the side. In French the same sentence "Les voitures ont vingt et un peint sur la portiere." would be ambiguous. "vingt et un" can generally have two interpretations, either 20 and 1 as in English, or 21. By common knowledge, the latter is the traditionnal interpretation except for special circimstances like the use of the word "respectivement" (respectively). This case is limited to the numbers between 20 and 80, ending in one, namely "vingt et un" (21), "trente et un" (31), "quarante et un" (41), "cinquante et un" (51), "soixante et un" (61), "soixante et onze" (71), and their compounds, e.g. "cent vingt et un" (121). So that the parsing of the conjunction of two numbers checks for such occurences, and prevents any further processing.

Interrogation

Instead of using auxiliary verbs like "do", French places in front of the question the universal "Est-ce-que..." (Is it so that...). When they start a sentence these two structures always introduce a question. ASK then picks up the noun phrase following these structures and treats them as an interrogation.

Both languages also use BE verb inversion to express interrogation, like in "Is Socrates mortal?" except that French employs pronouns "Est-il vivant?" (Is he alive?).

The basic interrogative sentence is similarly introduced by a wh-words like "quand" (when) or "combien" (how many) in both languages. However the use of the auxiliary is often quite different. For example "y a-t-il..." is constructed around a "have+PRESENT" auxiliary verb with a clitic inversion of "y", as opposed to the english equivalent "is there...", which is constructed around a "be+PRESENT" with a normal subject inversion.

DIAGNOSTICS

Let's consider a simple, but common example of diagnostics to explain the multilingual aspect of the answering mechanism of ASK. Suppose there isn't any relevant information in the data base about a class (e.g. girls). In this case, the procedure that handles this diagnostic, is handed a noun phrase and it must format an output that in English looks like:

"There be+PRESENT no " <noun_phrase> "."

In French the template would be:

"Il n'y avoir+PRESENT pas de " <phrase_nom> "."

When an output procedure is handed a variable (i.e. <noun_phrase>) like in this case, the interaction of this variable with the fixed part of the diagnostic (i.e. in double quotes) must be analyzed. Usually three cases arise.

First case, the variable doesn't trigger any morphological rule in any language and the two segments are said to be independant. The variable is then inserted, as is, in the diagnostic.

Second case, the variable triggers an identical rule in all languages. The variable is inserted in the diagnostic, and the morphological rule is called on the whole sentence.

Third case, the variable triggers a rule only in a specific language. The morphologic rule is applied only to the segment in that language. This is the case of our example. Each language requires a specific rule to be applied.

>Who is a girl?
There is no girl.

>Who are foreigners?
There are no foreigners.

>Qui est une fille?
Il n'y a pas de fille.

>Qui sont des etrangers?
Il n'y a pas d'etrangers.

In English, if the noun phrase is singular, the morphological transformation:

"be+PRESENT" --> "is"

is applied, otherwise it will use:

"be+PRESENT" --> "are"

making the verb agree with the noun.

In French, this problem does not arise since the gallicism "Il n'y a pas" is invariable, but if the noun phrase starts with a vowel, the morphological contraction:

"de " <phrase_nom> --> "d'" <phrase_nom>

is applied. This rule doesn't apply to english, of course.

Since the output routines in ASK are shared by all languages, provision has to be made in the code to allow this sort of manipulation. ASK keeps all the predetermined fragments of answers in a parallel table. A part of this table, relevant to our example, is shown below.

English table

```
 5: st:='There is none.';
 6: st:='There are none.';
 7: st:='There are no ';
 8: st:='There is no ';
 9: st:='';
10: st:='';
```

French table

```
 5: st:='Il n''y en a pas.';
 6: st:='Il n''y en a pas.';
 7: st:='Il n''y a pas ';
 8: st:='Il n''y a pas ';
 9: st:='d''';
10: st:='de ';
```

And the structure of the replies in the example above would be in either language:

```
diagnostic(8)+diagnostic(10)+<noun_phrase>+"."
```

and

```
diagnostic(7)+diagnostic(9)+<noun_phrase>+"."
```

where diagnostic(n) refers to the nth entry in the table and + to the concatenation operator.

To allow output in another language is a matter of translating all the different fragments of each table and adding new specific entries for that language. One advantage of this technique is that it makes it very easy to select in what language the answers should be made. It also allows for "en cours" changes of language.

BIBLIOGRAPHY

BARLOW, Frank, "The Feudal Kingdom of England, 1042-1216", London, New-York, Longmans, Green, 1955

CHOMSKY, Noam, "Syntactic Structures," Mouton & co, 1966

CHOMSKY, Noam, "Aspects of the Theory of Syntax," Cambridge, MIT Press, 1965

DUBOIS, Marguerite-Marie, "The Essentials of French Grammar," Librairie Larousse, 1960

KAYNE, Richard S., "French Syntax, the Transformational Cycle," MIT Press, 1975

LANGACKER, Ronald W., "Language and its Structure, some fundamental linguistic concepts," New-York, Harcourt, Bruce & World, 1968

WOUTAZ, Fernand, "Comment reconnaitre 30 bons vins," Hatier, Paris, 1978

APPENDIX I

Examples:

The following pages are examples of typical sessions with ASK-French. A data base is first built, and then the user querries it. The user's input is on the lines starting with a ">" character. Any input starting with a "*" is treated as a comment. ASK will often echo the input, adding parentheses to it, to indicate how it disambiguated the querry.

```
>*** A French wine data base
>*** Defining different classes:
>Cree les categories nommees vin,Alsace,Champagne,Bourgogne,Beaujolais
 Les nouvelles catégories suivantes ont été ajoutées:
   vin  Alsace  Champagne  Bourgogne  Beaujolais
>Cree les categories:Cotes-du-Rhone,Provence,Bordeaux,Muscadet
 Les nouvelles catégories suivantes ont été ajoutées:
   Cotes-du-Rhone  Provence  Bordeaux  Muscadet
>*** Checking the input
>categories?
 catégories d'individu:
 Beaujolais
 Bourgogne
 Champagne
 Alsace
 vin
 Muscadet
 Bordeaux
 Provence
 Cotes-du-Rhone
>Cree les individus:blanc, rose, rouge
 les nouveaux individus suivants ont été ajoutés:
   blanc  rose  rouge
>Cree un attribut du nom de couleur
 Le nouvel attribut d'individus couleur a été ajouté.
>Alsace, Champagne, Bourgogne et Beaujolais sont des vins.
 Alsace  Champagne  Bourgogne  Beaujolais
   ont été ajouté à la catégorie vins.
>Cotes-du-Rhone, Provence, Bordeaux et Muscadet sont des vins.
 Cotes-du-Rhone  Provence  Bordeaux  Muscadet
   ont été ajouté à la catégorie vins.
```

```
>La couleur des Champagnes est blanc.
 blanc a été ajouté en tant que couleur de Champagnes.
>La couleur des Beaujolais est rouge.
 rouge a été ajouté en tant que couleur de Beaujolais.
>La couleur des Cotes-du-Rhones est rose.
 rose a été ajouté en tant que couleur de Cotes-du-Rhones.
>La couleur des Provence est rose.
 rose a été ajouté en tant que couleur de Provence.
>La couleur des Muscadet est blanc.
 blanc a été ajouté en tant que couleur de Muscadet.
>
>*** Defining individual entities:
>Cree des individus nommes Sylvaner,Pinot noir,Riesling
 Les nouveaux individus suivants ont été ajoutés:
   Sylvaner  Pinot noir  Riesling
>individus:Tokay,Muscat,Gewurtztraminer
 Les nouveaux individus suivants ont été ajoutés:
   Tokay  Muscat  Gewurtztraminer
>individus?
 rouge
 rose
 blanc
 Riesling
 Pinot noir
 Sylvaner
 Gewurtztraminer
 Muscat
 Tokay
>*** Storing the individuals in classes:
>Sylvaner,Pinot noir et Riesling sont des Alsaces.
 Sylvaner  Pinot noir  Riesling
   ont été ajouté à la catégorie Alsaces.
>*** Linking individuals through attributes:
>La couleur des Tokay, Sylvaner et Riesling est blanc.
 blanc a été ajouté en tant que couleur de Tokay.
 blanc a été ajouté en tant que couleur de Sylvaner.
 blanc a été ajouté en tant que couleur de Riesling.
>La couleur du Pinot noir est rose.
 rose a été ajouté en tant que couleur de Pinot noir.
>Tokay, Muscat et Gewurtztraminer sont des Alsaces.
 Tokay  Muscat  Gewurtztraminer
   ont été ajouté à la catégorie Alsaces.
>La couleur des Muscat et Gewurtztraminer est blanc.
 blanc a été ajouté en tant que couleur de Muscat.
 blanc a été ajouté en tant que couleur de Gewurtztraminer.
```

```
>couleur de chaque Alsace?
 Alsace              couleur
 Riesling            blanc
 Pinot noir          rose
 Sylvaner            blanc
 Gewurtztraminer     blanc
 Muscat              blanc
 Tokay               blanc
>
>individu:Dom Perignon, Veuve Cliquot
 Les nouveaux individus suivants ont été ajoutés:
   Dom Perignon  Veuve Cliquot
>Dom Perignon et Veuve Cliquot sont des Champagnes.
 Dom Perignon  Veuve Cliquot
   ont été ajouté à la catégorie Champagnes.
>
>individu:Gevrey-Chambertin, Nuits-Saint-Georges
 Les nouveaux individus suivants ont été ajoutés:
   Gevrey-Chambertin  Nuits-Saint-Georges
>individu:Pommard,Volnay,Meursault,Pouilly-Fuisse
 Les nouveaux individus suivants ont été ajoutés:
   Pommard  Volnay  Meursault  Pouilly-Fuisse
>individu:Macon, Clos-Vougeot
 Les nouveaux individus suivants ont été ajoutés:
   Macon  Clos-Vougeot
>Gevrey-Chambertin et  Nuits-Saint-Georges sont des Bourgognes.
 Gevrey-Chambertin  Nuits-Saint-Georges
   ont été ajouté à la catégorie Bourgognes.
>Couleur de Gevrey-Chambertin, Nuits-Saint-Georges et Clos-Vougeot est rouge.
 rouge a été ajouté en tant que couleur de Gevrey-Chambertin.
 rouge a été ajouté en tant que couleur de Nuits-Saint-Georges.
 rouge a été ajouté en tant que couleur de Clos-Vougeot.
>Pommard,Volnay,Meursault et Pouilly-Fuisse sont des Bourgognes.
 Pommard  Volnay  Meursault  Pouilly-Fuisse
   ont été ajouté à la catégorie Bourgognes.
>La couleur des Volnay, Macon, Pommard, Meursault et Pouilly-Fuisse est blanc.
 blanc a été ajouté en tant que couleur de Volnay.
 blanc a été ajouté en tant que couleur de Macon.
 blanc a été ajouté en tant que couleur de Pommard.
 blanc a été ajouté en tant que couleur de Meursault.
 blanc a été ajouté en tant que couleur de Pouilly-Fuisse.
>Clos-Vougeot et Macon sont des Bourgognes.
 Clos-Vougeot  Macon
   ont été ajouté à la catégorie Bourgognes.
```

```
>Quels sont les Bourgognes?
 Nuits-Saint-Georges
 Gevrey-Chambertin
 Pouilly-Fuisse
 Meursault
 Volnay
 Pommard
 Clos-Vougeot
 Macon
>Quelle est la couleur des Bourgognes?
 rouge
 blanc
>*** An example of echoing the input to clarify the result
>Quels sont les Bourgognes rouges?
 Bourgognes (couleur) rouges
 Nuits-Saint-Georges
 Gevrey-Chambertin
 Clos-Vougeot
>
!individu:Saint-Amour,Julienas,Fleurie,Chenas
 Les nouveaux individus suivants ont été ajoutés:
   Saint-Amour  Julienas  Fleurie  Chenas
>individu:Moulin-a-Vent,Chiroubles,Morgon
 Les nouveaux individus suivants ont été ajoutés:
   Moulin-a-Vent  Chiroubles  Morgon
>individu:Cote-de-Brouilly et Brouilly
 Les nouveaux individus suivants ont été ajoutés:
   Cote-de-Brouilly  Brouilly
>Saint-Amour,Julienas,Fleurie et Chenas sont des Beaujolais.
 Saint-Amour  Julienas  Fleurie  Chenas
   ont été ajouté à la catégorie Beaujolais.
>Moulin-a-Vent,Chiroubles et Morgon sont des Beaujolais.
 Moulin-a-Vent  Chiroubles  Morgon
   ont été ajouté à la catégorie Beaujolais.
>Cote-de-Brouilly et Brouilly sont des Beaujolais.
 Cote-de-Brouilly  Brouilly
   ont été ajouté à la catégorie Beaujolais.
>Les Beaujolais?
 Chenas
 Fleurie
 Julienas
 Saint-Amour
 Morgon
 Chiroubles
 Moulin-a-Vent
 Brouilly
 Cote-de-Brouilly
>
```

```
>individu:Saint_Peray,Cote-Rotie,Crozes-Hermitage
 Les nouveaux individus suivants ont été ajoutés:
   Saint_Peray  Cote-Rotie  Crozes-Hermitage
>individu:Chateauneuf-du-Pape,Gigondas
 Les nouveaux individus suivants ont été ajoutés:
   Chateauneuf-du-Pape  Gigondas
>Saint_Peray,Cote-Rotie et Crozes-Hermitage sont des Cotes-du-Rhone.
 Saint_Peray  Cote-Rotie  Crozes-Hermitage
   ont été ajouté à la catégorie Cotes-du-Rhone.
>Chateauneuf-du-Pape et Gigondas sont des Cotes-du-Rhone.
 Chateauneuf-du-Pape  Gigondas
   ont été ajouté à la catégorie Cotes-du-Rhone.
>Enumere les Cotes-du-Rhone.
 enumere Cotes-du-Rhone?
 Crozes-Hermitage
 Cote-Rotie
 Saint_Peray
 Gigondas
 Chateauneuf-du-Pape
>
>individu:Bandol,Cassis,Bellet,Cotes-de-Provence
 Les nouveaux individus suivants ont été ajoutés:
   Bandol  Cassis  Bellet  Cotes-de-Provence
>Bandol,Cassis,Bellet et Cotes-de-Provence sont des Provences.
 Bandol  Cassis  Bellet  Cotes-de-Provence
   ont été ajouté à la catégorie Provences.
>Provence?.
 Cotes-de-Provence
 Bellet
 Cassis
 Bandol
>
>individu:Entre-deux-Mers,Graves de Vayres,Medoc
 Les nouveaux individus suivants ont été ajoutés:
   Entre-deux-Mers  Graves de Vayres  Medoc
>individu:Haut Medoc,Margaux,Saint-Estephe
 Les nouveaux individus suivants ont été ajoutés:
   Haut Medoc  Margaux  Saint-Estephe
>individu:Saint-Emilion,Pomerol,Sauternes
 Les nouveaux individus suivants ont été ajoutés:
   Saint-Emilion  Pomerol  Sauternes
>Entre-deux-Mers,Graves de Vayres et Medoc sont des Bordeaux.
 Entre-deux-Mers  Graves de Vayres  Medoc
   ont été ajouté à la catégorie Bordeaux.
>La couleur des Entre-deux-Mers et Graves de Vayres est blanc.
 blanc a été ajouté en tant que couleur d'Entre-deux-Mers.
 blanc a été ajouté en tant que couleur de Graves de Vayres.
```

```
>Haut Medoc,Margaux et Saint-Estephe sont des Bordeaux.
 Haut Medoc  Margaux  Saint-Estephe
   ont été ajouté à la catégorie Bordeaux.
>Saint-Emilion,Pomerol et Sauternes sont des Bordeaux.
 Saint-Emilion  Pomerol  Sauternes
   ont été ajouté à la catégorie Bordeaux.
>La couleur des Medoc, Haut Medoc, Margaux et Saint-Estephe est rouge.
 rouge a été ajouté en tant que couleur de Medoc.
 rouge a été ajouté en tant que couleur d'Haut Medoc.
 rouge a été ajouté en tant que couleur de Margaux.
 rouge a été ajouté en tant que couleur de Saint-Estephe.
>La couleur des Saint-Emilion,Pomerol et Sauternes est rouge.
 rouge a été ajouté en tant que couleur de Saint-Emilion.
 rouge a été ajouté en tant que couleur de Pomerol.
 rouge a été ajouté en tant que couleur de Sauternes.
>Enumere les Bordeaux rouges.
 Bordeaux (couleur) rouges
 enumere Bordeaux rouges?
 Medoc
 Saint-Estephe
 Margaux
 Haut Medoc
 Sauternes
 Pomerol
 Saint-Emilion
>Enumere les Bordeaux blancs.
 Bordeaux (couleur) blancs
 enumere Bordeaux blancs?
 Graves de Vayres
 Entre-deux-Mers
>
>*** Defining a number attribute
>unite:franc
 La nouvelle unité franc a été ajoutée.
>attribut numerique:prix avec comme unite le franc
 Veuillez donner les deux adjectifs associés
 à prix (comme "grand" et "petit" sont
 associés à "taille"); l'adjectif maximisateur
 (tel que "grand") avant l'adjectif
 minimisateur (tel que "petit"):cher - bon marche
 Les adjectifs cher et bon marche ont été
 rajoutés, associés à prix.
 Le nouvel attribut numérique prix a été ajouté.
>Le prix du Pinot noir est 17.80.
 17.80 a été ajouté en tant que prix de Pinot noir.
>Le prix du Riesling est 23.75.
 23.75 a été ajouté en tant que prix de Riesling.
>Le prix du Tokay est 21.
 21 a été ajouté en tant que prix de Tokay.
```

```
>Le prix du Dom Perignon est 378.50.
 378.50 a été ajouté en tant que prix de Dom Perignon.
>Le prix du Nuits-Saint-Georges est 43.20.
 43.20 a été ajouté en tant que prix de Nuits-Saint-Georges.
>Le prix du Volnay est 31.05.
 31.05 a été ajouté en tant que prix de Volnay.
>Le prix du Pouilly-Fuisse est 72.00.
 72.00 a été ajouté en tant que prix de Pouilly-Fuisse.
>Le prix du Juliennas est 29.80.
 29.80 a été ajouté en tant que prix de Julienas.
>Le prix du Chiroubles est 25.
 25 a été ajouté en tant que prix de Chiroubles.
>Le prix du Brouilly est 37.65.
 37.65 a été ajouté en tant que prix de Brouilly.
>Le prix du Crozes-Hermitage est 18.85.
 18.85 a été ajouté en tant que prix de Crozes-Hermitage.
>Le prix du Chateauneuf-du-Pape est 24.70.
 24.70 a été ajouté en tant que prix de Chateauneuf-du-Pape.
>Le prix du Gigondas est 13.20.
 13.20 a été ajouté en tant que prix de Gigondas.
>Le prix du Bellet est 42.75.
 42.75 a été ajouté en tant que prix de Bellet.
>Le prix du Cotes-de-Provence est 9.50.
 9.50 a été ajouté en tant que prix de Cotes-de-Provence.
>Le prix de l'Entre-deux-Mers est 42.50.
 42.50 a été ajouté en tant que prix d'Entre-deux-Mers.
>Le prix du Medoc est 95.05.
 95.05 a été ajouté en tant que prix de Medoc.
>Le prix du Saint-Emilion est 80.00.
 80.00 a été ajouté en tant que prix de Saint-Emilion.
>Le prix du Sauternes est 65.40.
 65.40 a été ajouté en tant que prix de Sauternes.
>Quel est le prix de chaque Alsace?
 Alsace          prix
                 franc
 Riesling          23.75
 Pinot noir        17.8
 Tokay             21.
>19.90 est le prix du Pinot noir.
 17.8 a été remplacé par 19.90 en tant que prix de Pinot noir.
>Quel est le prix du Pinot noir?
 19.9 franc
>
```

```
>*** Defining a number relation
>relation numerique:annee exceptionelle
 Veuillez donner les deux adjectifs associés
 à annee exceptionelle (comme "grand" et "petit" sont
 associés à "taille"); l'adjectif maximisateur
 (tel que "grand") avant l'adjectif
 minimisateur (tel que "petit"):
 La nouvelle relation numérique annee exceptionelle a été ajoutée.
>*** Use definitional capacities to resolve compound plurals:
>definition:annees exceptionelle:annee exceptionelle
 Défini.
>1928 et 1976 sont des annee exceptionelles pour les Alsaces.
 1928 a été ajouté en tant qu'annee exceptionelle d'Alsaces.
 1976 a été ajouté en tant qu'annee exceptionelle d'Alsaces.
>1928 ,1934 et 1966 furent des annee exceptionelles pour les Champagnes.
 1928 a été ajouté en tant qu'annee exceptionelle de Champagnes.
 1934 a été ajouté en tant qu'annee exceptionelle de Champagnes.
 1966 a été ajouté en tant qu'annee exceptionelle de Champagnes.
>1961 et 1969 sont des annee exceptionelles pour les Bourgognes.
 1961 a été ajouté en tant qu'annee exceptionelle de Bourgognes.
 1969 a été ajouté en tant qu'annee exceptionelle de Bourgognes.
>1929 est une annee exceptionelle pour les Bourgognes.
 1929 a été ajouté en tant qu'annee exceptionelle de Bourgognes.
>Les annee exceptionelles pour les Beaujolais furent 1926, 1947 et 1953.
 1926 a été ajouté en tant qu'annee exceptionelle de Beaujolais.
 1947 a été ajouté en tant qu'annee exceptionelle de Beaujolais.
 1953 a été ajouté en tant qu'annee exceptionelle de Beaujolais.
>L'annee exceptionelle des Cotes-du-Rhone fut 1966.
 1966 a été ajouté en tant qu'annee exceptionelle de Cotes-du-Rhone.
>Les annee exceptionelles des Bordeaux rouges sont 1928 et 1970.
 1928 a été ajouté en tant qu'annee exceptionelle de Medoc.
 1928 a été ajouté en tant qu'annee exceptionelle de Saint-Estephe.
 1928 a été ajouté en tant qu'annee exceptionelle de Margaux.
 1928 a été ajouté en tant qu'annee exceptionelle d'Haut Medoc.
 1928 a été ajouté en tant qu'annee exceptionelle de Sauternes.
 1928 a été ajouté en tant qu'annee exceptionelle de Pomerol.
 1928 a été ajouté en tant qu'annee exceptionelle de Saint-Emilion.
 1970 a été ajouté en tant qu'annee exceptionelle de Medoc.
 1970 a été ajouté en tant qu'annee exceptionelle de Saint-Estephe.
 1970 a été ajouté en tant qu'annee exceptionelle de Margaux.
 1970 a été ajouté en tant qu'annee exceptionelle d'Haut Medoc.
 1970 a été ajouté en tant qu'annee exceptionelle de Sauternes.
 1970 a été ajouté en tant qu'annee exceptionelle de Pomerol.
 1970 a été ajouté en tant qu'annee exceptionelle de Saint-Emilion.
```

```
>Les annee exceptionelles des Bordeaux blancs sont 1945 et 1967.
 1945 a été ajouté en tant qu'annee exceptionelle de Graves de Vayres.
 1945 a été ajouté en tant qu'annee exceptionelle d'Entre-deux-Mers.
 1967 a été ajouté en tant qu'annee exceptionelle de Graves de Vayres.
 1967 a été ajouté en tant qu'annee exceptionelle d'Entre-deux-Mers.
>annee exceptionelle des Medocs?
 1928
 1970
>
>
>*** Now show the state of the current data base
>Enumere la couleur et le prix de chaque vin.
 Votre réponse comprend 45 lignes.
>Lesquelles voulez-vous? dix sept
```

vin	couleur	prix franc
Riesling	blanc	25.75
Pinot noir	rose	19.9
Sylvaner	blanc	---
Gewurtztraminer	blanc	---
Muscat	blanc	---
Tokay	blanc	21.
Veuve Cliquot	blanc	---
Dom Perignon	blanc	378.5
Nuits-Saint-Georges	rouge	43.2
Gevrey-Chambertin	rouge	---
Pouilly-Fuisse	blanc	72.
Meursault	blanc	---
Volnay	blanc	31.05
Pommard	blanc	---
Clos-Vougeot	rouge	---

```
>Quelles lignes voulez-vous? le reste
```

Macon	blanc	---
Chenas	rouge	---
Fleurie	rouge	---
Julienas	rouge	29.8
Saint-Amour	rouge	---
Morgon	rouge	---
Chiroubles	rouge	25.
Moulin-a-Vent	rouge	---
Brouilly	rouge	37.65
Cote-de-Brouilly	rouge	---
Crozes-Hermitage	rose	18.85
Cote-Rotie	rose	---
Saint Peray	rose	---
Gigondas	rose	13.2
Chateauneuf-du-Pape	rose	24.7
Cotes-de-Provence	rose	9.5

```
Bellet                  rose        42.75
Cassis                  rose        ---
Bandol                  rose        ---
Medoc                   rouge       95.05
Graves de Vayres        blanc       ---
Entre-deux-Mers         blanc       42.5
Saint-Estephe           rouge       ---
Margaux                 rouge       ---
Haut Medoc              rouge       ---
Sauternes               rouge       45.4
Pomerol                 rouge       ---
Saint-Emilion           rouge       80.
)Quelles lignes voulez-vous? )
>Quelle est l'annee exceptionelle de chaque vin?
 Votre réponse comprend 93 lignes.
>Lesquelles voulez-vous? toutes
 vin                    annee exceptionelle
 Riesling                               1928
                                        1976
 Pinot noir                             1928
                                        1976
 Sylvaner                               1928
                                        1976
 Gewurtztraminer                        1928
                                        1976
 Muscat                                 1928
                                        1976
 Tokay                                  1928
                                        1976
 Veuve Cliquot                          1928
                                        1934
                                        1966
 Dom Perignon                           1928
                                        1934
                                        1966
 Nuits-Saint-Georges                    1929
                                        1961
                                        1969
 Gevrey-Chambertin                      1929
                                        1961
                                        1969
 Pouilly-Fuisse                         1929
                                        1961
                                        1969
 Meursault                              1929
                                        1961
                                        1969
```

Volnay	1929
	1961
	1969
Pommard	1929
	1961
	1969
Clos-Vougeot	1929
	1961
	1969
Macon	1929
	1961
	1969
Chenas	1926
	1947
	1953
Fleurie	1926
	1947
	1953
Julienas	1926
	1947
	1953
Saint-Amour	1926
	1947
	1953
Morgon	1926
	1947
	1953
Chiroubles	1926
	1947
	1953
Moulin-a-Vent	1926
	1947
	1953
Brouilly	1926
	1947
	1953
Cote-de-Brouilly	1926
	1947
	1953
Crozes-Hermitage	1966
Cote-Rotie	1966
Saint Peray	1966
Gigondas	1966
Chateauneuf-du-Pape	1966
Medoc	1928
	1970

```
Graves de Vayres                    1945
                                    1967
Entre-deux-Mers                     1945
                                    1967
Saint-Estephe                       1928
                                    1970
Margaux                             1928
                                    1970
Haut Medoc                          1928
                                    1970
Sauternes                           1928
                                    1970
Pomerol                             1928
                                    1970
Saint-Emilion                       1928
                                    1970
>*** Now check more specific details:
>Quels sont les vins blancs?
 vins (couleur) blancs
 Riesling
 Sylvaner
 Gewurtztraminer
 Muscat
 Tokay
 Veuve Cliquot
 Dom Perignon
 Pouilly-Fuisse
 Meursault
 Volnay
 Pommard
 Macon
 Graves de Vayres
 Entre-deux-Mers
>Quelle est la couleur du Bellet?
 rosé
>Quelles sont les annees exceptionelles des vins roses?
 1928
 1966
 1976
>Est-ce-que le Pinot noir est un Champagne?
 non
>Le Pomerol est un Bordeaux rouge?
 Bordeaux (couleur) rouge
 oui
```

```
>definition:dollar: 8.6 francs
 Défini.
>Quel est le prix des Bourgognes et des Champagnes?
 quel est le prix (des (Bourgognes et des Champagnes))?
 Bourgognes et Champagnes     prix
                             franc
 Bourgogne                     31.05
                               43.2
                               72.
 Champagne                    378.5
>Quel est le prix de chaque Bourgogne en dollar?
 Bourgogne              prix de chaque Bourgogne
                                          dollar
 Nuits-Saint-Georges                 5.023255813953487
 Pouilly-Fuisse                      8.372093023255814
 Volnay                              3.610465116279069
>Est ce que le prix du Volnay est plus eleve que le prix du Chiroubles?
 oui
>Y a t'il des Muscadets?
 non
>*** Apply statistical operators on the data base:
>Quelle est la moyenne du prix des Beaujolais?
 30.816666666666667
```

APPENDIX II

French Syntax

ASK Syntax is composed essentially of a set of grammatical rewrite rules. The ASK format is as follows: A rule is initiated by the keyword "RULE", anything following it on the same line is considered to be a comment which appears on the screen of the computer when the system assimilates the whole syntax. The following lines are composed of a left hand side and a right hand side seperated by the symbols "=>". Basically, the right hand side gets rewritten into the left hand side's format. Both of the sides can be composed of two elements: literals which are enclosed between double quotes and are taken at face value, and parts of speech which are enclosed between a "<" and a ">". The parts of speech can require certain features to be set or reset. This is specified in a list placed after the name of the part of speech and preceded with a ":". A plus or minus sign placed in front of the name of the feature specifies if the feature should be on or off. The third part of the rule is a line starting with either LEX, SYN, PRE or POST. This specifies at what time the rule should be applied. LEX means the rule should be added in the lexicon, SYN means the rule should be applied at syntax time, and PRE or POST means it should be applied at semantic time, before or after, respectively, the preprocessor is called. After a SYN, PRE or POST comes the name of the semantic procedure that is associated with the rule and which is called on the resulting parsing of the rule. Any line starting with a "*" is a comment.

```
[version: 29-Jul-1983                                    FR:LEX.SYN
FR:LEX.SYN          The Syntax of ASK French
                          Basic Lexicon                          ]

RULE "entite en cours"
<mot:+cur_ent> => "entite_en_cours"
LEX 6

RULE "le"
<article_defini:+mas> => "le"
```

```
LEX 0

RULE "le"
<article_defini:+fem> => "la"
LEX 0

RULE "les"
<article_defini:+plf> => "les"
LEX 0

RULE "l'"
<article_defini:+voy> => "l'"
LEX 0

RULE "un"
<article_indefini:+mas> => "un"
LEX 0

RULE "une"
<article_indefini:+fem> => "une"
LEX 0

RULE "des"
<article_indefini:+plf> => "des"
LEX 0

RULE "nombre"
<mot:+num_of> => "nombre"
LEX 0

RULE "compte"
<mot:+count> => "compte"
LEX 0

RULE "reciproque"
<mot:+converse> => "reciproque"
LEX 0

RULE le quantificatif "tout"
<quantificatif:+plf+sif+otf+mas> => "tout"
LEX 36
```

```
RULE le quantificatif "toute"
(quantificatif:+plf+sif+otf+fem) => "toute"
LEX 36

RULE le quantificatif "tous"
(quantificatif:+plf+otf+mas+plf) => "tous"
LEX 36

RULE le quantificatif "toutes"
(quantificatif:+plf+otf+fem+plf) => "toutes"
LEX 36

RULE le quantificatif "les deux"
(quantificatif:+plf+otf) => "les deux"
LEX 38

RULE le quantificatif "aucun"
(quantificatif:+plf+sif+mas) => "aucun"
LEX 24

RULE le quantificatif "aucune"
(quantificatif:+plf+sif+fem) => "aucune"
LEX 24

RULE le quantificatif "quelque"
(quantificatif:+plf+sif+otf) => "quelque"
LEX 32

RULE le quantificatif "quelques"
(quantificatif:+plf+sif+otf) => "quelques"
LEX 32

RULE le quantificatif "l'un ou l'autre"
(quantificatif:+sif+otf) => "l'un ou l'autre"
LEX 32

RULE le quantificatif "l'une ou l'autre"
(quantificatif:+sif+otf+fem) => "l'une ou l'autre"
LEX 32

RULE le quantificatif "chaque"
(quantificatif:+sif+otf) => "chaque"
LEX 31

RULE le quantificatif "chacun"
(quantificatif:+sif+otf) => "chacun"
LEX 31
```

```
RULE le quantificatif "quel"
(quantificatif:+whf+sif+mas) => "quel"
LEX 30

RULE le quantificatif "quelle"
(quantificatif:+whf+sif+fem) => "quelle"
LEX 30

RULE le quantificatif "quel"
(quantificatif:+whf+plf+sif+mas) => "quels"
LEX 30

RULE le quantificatif "quel"
(quantificatif:+whf+plf+sif+fem) => "quelles"
LEX 30

RULE le quantificatif "combien"
(quantificatif:+whf+plf+otf) => "combien"
LEX 27

RULE le quantificatif "au moins"
(quantificatif:+num+plf+sif+otf) => "au moins"
LEX 21

RULE le quantificatif "au plus"
(quantificatif:+num+plf+sif+otf) => "au plus"
LEX 22

RULE le quantificatif "exactement"
(quantificatif:+num+plf+sif+otf) => "exactement"
LEX 23

RULE le quantificatif "tout sauf"
(quantificatif:+num+plf+sif+otf) => "tout sauf"
LEX 24

RULE le quantificatif "tous sauf"
(quantificatif:+num+plf+sif+otf) => "tous sauf"
LEX 24

RULE le quantificatif "plusieurs"
(quantificatif:+plf+otf) => "plusieurs"
LEX 33

RULE le quantificatif "tout les"
(quantificatif:+plf) => "tout les"
LEX 36
```

```
RULE le quantificatif "tous les"
(quantificatif:+plf) => "tous les"
LEX 36

RULE le quantificatif "plus que"
(quantificatif:+num+plf+sif+otf) => "plus que"
LEX 34

RULE le quantificatif "plus"
(quantificatif:+num+plf+otf) => "plus"
LEX 34

RULE le quantificatif "moins que"
(quantificatif:+num+plf+sif+otf) => "moins que"
LEX 35

RULE le quantificatif "moins"
(quantificatif:+num+plf+sif+otf) => "moins"
LEX 35

RULE le pronom relatif  "qui"
(pronom_relatif:+who) => "qui"
LEX 0

RULE le pronom relatif "quoi"
(pronom_relatif:+what) => "quoi"
LEX 0

RULE le pronom relatif "quel"
(pronom_relatif:+what) => "quel"
LEX 0

RULE le pronom relatif "quels"
(pronom_relatif:+what) => "quels"
LEX 0

RULE le pronom relatif "quelle"
(pronom_relatif:+what) => "quelle"
LEX 0

RULE le pronom relatif "quelles"
(pronom_relatif:+what) => "quelles"
LEX 0

RULE le pronom relatif  "que"
(pronom_relatif:+that) => "que"
LEX 0
```

```
RULE le pronom relatif  "qu'"
(pronom_relatif:+vol+that) => "qu'"
LEX 0

RULE le pronom relatif  "lequel"
(pronom_relatif:+which+mas) => "lequel"
LEX 0

RULE le pronom relatif  "laquelle"
(pronom_relatif:+which+fem) => "laquelle"
LEX 0

RULE le pronom relatif  "lesquels"
(pronom_relatif:+which+mas+plf) => "lesquels"
LEX 0

RULE le pronom relatif  "lesquelles"
(pronom_relatif:+which+fem+plf) => "lesquelles"
LEX 0

RULE le pronom relatif  "auquel"
(pronom_relatif:+which+mas+au) => "auquel"
LEX 0

RULE le pronom relatif  "auxquels"
(pronom_relatif:+which+mas+plf+au) => "auxquels"
LEX 0

RULE le pronom relatif  "auxquelles"
(pronom_relatif:+which+fem+plf+au) => "auxquelles"
LEX 0

RULE le pronom relatif  "a laquelle"
(pronom_relatif:+which+fem+au) => "a laquelle"
LEX 0

RULE le pronom relatif  "duquel"
(pronom_relatif:+which+mas+de) => "duquel"
LEX 0

RULE le pronom relatif  "desquels"
(pronom_relatif:+which+mas+plf+de) => "desquels"
LEX 0

RULE le pronom relatif  "de laquelle"
(pronom_relatif:+which+fem+de) => "de laquelle"
LEX 0
```

```
RULE le pronom relatif  "desquelles"
(pronom_relatif:+which+fem+plf+de) => "desquelles"
LEX 0

RULE le pronom relatif  "dont"
(pronom_relatif:+whose) => "dont"
LEX 0

RULE "contre"
(mot:+versus) => "contre"
LEX 0

RULE "oppose a"
(mot:+versus) => "oppose a"
LEX 0

RULE "il"
(pronom:+anf+dtf+il) => "il"
LEX 0

RULE "elle"
(pronom:+anf+dtf) => "elle"
LEX 0

RULE "son"
(pronom:+anf+dtf+pof) => "son"
LEX 0

RULE "lui"
(pronom:+anf+dtf) => "lui"
LEX 0

RULE "sa"
(pronom:+anf+dtf+pof) => "sa"
LEX 0

RULE "c'"
(pronom:+dtf+vol) => "c'"
LEX 0

RULE "ca"
(pronom:+dtf) => "ca"
LEX 0

RULE "ils"
(pronom:+anf+dtf+plf) => "ils"
LEX 0
```

```
RULE "leur"
(pronom:+anf+dtf+plf+pof) => "leur"
LEX 0

RULE "leur"
(pronom:+anf+dtf+plf) => "leur"
LEX 0

RULE "celui-ci"
(pronom:+dtf+mas) => "celui-ci"
LEX 0

RULE "celle-ci"
(pronom:+dtf+fem) => "celle-ci"
LEX 0

RULE "celle-la"
(pronom:+dtf+fem) => "celle-la"
LEX 0

RULE "celui-la"
(pronom:+dtf+mas) => "celui-la"
LEX 0

RULE "celles"
(pronom:+fem) => "celles"
LEX 0

RULE "ceux"
(pronom:+mas) => "ceux"
LEX 0

RULE "ce" PN
(pronom:+NP) => "ce"
LEX 0

RULE "autre"
(pronom:+NP) => "autre"
LEX 0

RULE "un autre"
(pronom:+NP+mas) => "un autre"
LEX 0

RULE "une autre"
(pronom:+NP+fem) => "une autre"
LEX 0
```

```
RULE "celui de"
(pronom:+NP+att+mas) => "celui de"
LEX 0

RULE "celle de"
(pronom:+NP+att+fem) => "celle de"
LEX 0

RULE "ceux de"
(pronom:+NP+att+mas+plf) => "ceux de"
LEX 0

RULE "celles de"
(pronom:+NP+att+fem+plf) => "celles de"
LEX 0

RULE "meme de"
(pronom:+NP+att) => "meme de"
LEX 0


**** Verbes

RULE "est"
(copule:+prim+fsi+is) => "est"
LEX 1

RULE "sont"
(copule:+prim+is) => "sont"
LEX 2

RULE "etre"
(copule:+prim+fif+is) => "etre"
LEX 3

RULE "etait"
(copule:+prim+fsi+is) => "etait"
LEX 4

RULE "etaient"
(copule:+prim+is) => "etaient"
LEX 5

RULE "fut"
(copule:+prim+fsi+is) => "fut"
*** flag should be 4, but too many features
LEX 1
```

```
RULE "furent"
(coqule:+prim+is) => "furent"
*** flag should be 5, but too many features
LEX 2

RULE "ete"
(coqule:+prim+fpp+is) => "ete"
LEX 6

RULE "ont"
(conule:+prim+fsi+have+plf) => "ont"
LEX 7

RULE "avoir"
(conule:+prim+have) => "avoir"
LEX 8

RULE "avait"
(conule:+prim+fpp+have) => "avait"
LEX 9

RULE l'interrogation "est ce que"
(verbe_aux:+does) => "est ce que"
LEX 0

RULE l'interrogation "est ce qu'"
(verbe_aux:+does+vol) => "est ce qu'"
LEX 0

RULE l'interrogation "est-ce que"
(verbe_aux:+does) => "est-ce que"
LEX 0

RULE l'interrogation "est-ce qu'"
(verbe_aux:+does+vol) => "est-ce qu'"
LEX 0

RULE l'interrogation "est-ce-que"
(verbe_aux:+does) => "est-ce-que"
LEX 0

RULE l'interrogation "est-ce-qu'"
(verbe_aux:+does+vol) => "est-ce-qu'"
LEX 0

RULE le mot interrogatif "qui"
(mot_inter:+anim) => "qui"
LEX 0
```

```
RULE le mot interrogatif "quelle"
<mot_inter:+anim> => "quelle"
LEX 0

RULE le mot interrogatif "quel"
<mot_inter:+mas> => "quel"
LEX 0

RULE le mot interrogatif "quelles"
<mot_inter:+fem+plf> => "quelles"
LEX 0

RULE le mot interrogatif "quels"
<mot_inter:+mas+plf> => "quels"
LEX 0

RULE le mot interrogatif "qu'est-ce qu'"
<mot_inter:+vol> => "qu'est-ce qu'"
LEX 0

RULE le mot interrogatif "qu'est-ce qui"
<mot_inter> => "qu'est-ce qui"
LEX 0

RULE le mot interrogatif "qu'est-ce que"
<mot_inter> => "qu'est-ce que"
LEX 0

RULE le mot interrogatif "combien"
<mot_inter:+how many> => "combien"
LEX 0

RULE "pas"
<adverbe:+not> => "pas"
LEX 0

RULE "n'"
<adverbe:+ne> => "n'"
LEX 0

RULE "ne"
<adverbe:+ne> => "ne"
LEX 0

RULE "y"
<adverbe:+there> => "y"
LEX 0
```

```
RULE "seulement"
(adverbe:+only) => "seulement"
LEX 0

RULE "seul"
(adverbe:+only) => "seul"
LEX 0

RULE "unique"
(adverbe:+only) => "unique"
LEX 0

RULE "uniquement"
(adverbe:+only) => "uniquement"
LEX 0

RULE "definition:"
(mot_nouv:+init+def) => "definition:"
LEX 0

RULE "verbe:"
(mot_nouv:+init+verb_def) => "verbe:"
LEX 0

RULE "nom"
(mot:+name) => "nom"
LEX 0

RULE "membre"
(mot:+name) => "membre"
LEX 0

RULE "que sait-on sur"
(sait_on) => "que sait-on sur"
LEX 0

RULE "que sait-on a propos de"
(sait_on) => "que sait-on a propos de"
LEX 0

RULE "que sait-on a propos d'"
(sait_on) => "que sait-on a propos d'"
LEX 0

RULE "que sait on sur"
(sait_on) => "que sait on sur"
LEX 0
```

```
RULE "que sait on a propos de"
<sait_on> => "que sait on a propos de"
LEX 0

RULE "que sait on a propos d'"
<sait_on> => "que sait on a propos d'"
LEX 0

RULE "decrit"
<sait_on> => "decrit"
LEX 0

RULE "decrivez"
<sait_on> => "decrivez"
LEX 0

RULE "que sais-tu sur"
<sait_on> => "que sais-tu sur"
LEX 0

RULE "que sais-tu a propos de"
<sait_on> => "que sais-tu a propos de"
LEX 0

RULE "que sais-tu a propos d'"
<sait_on> => "que sais-tu a propos d'"
LEX 0

RULE "que sais tu sur"
<sait_on> => "que sais tu sur"
LEX 0

RULE "que sais tu a propos de"
<sait_on> => "que sais tu a propos de"
LEX 0

RULE "que sais tu a propos d'"
<sait_on> => "que sais tu a propos d'"
LEX 0

RULE "que savez-vous sur"
<sait_on> => "que savez-vous sur"
LEX 0

RULE "que savez-vous a propos de"
<sait_on> => "que savez-vous a propos de"
LEX 0
```

```
RULE "que savez vous a propos d'"
(sait_on) => "que savez-vous a propos d'"
LEX 0

RULE "que savez vous sur"
(sait_on) => "que savez vous sur"
LEX 0

RULE "que savez vous a propos de"
(sait_on) => "que savez vous a propos de"
LEX 0

RULE "que savez vous a propos d'"
(sait_on) => "que savez vous a propos d'"
LEX 0

RULE "enumere"
(mot:+list) => "enumere"
LEX 0

RULE "enumerez"
(mot:+list) => "enumerez"
LEX 0

***** Regles lexicologiques d'arithmetique

RULE plus binaire
(binary_op:+asf) => "plus"
LEX 1

RULE moins binaire
(binary_op:+asf) => "moins"
LEX 2

RULE soustraction: retire a
(binary_op:+asf) => "retire a"
LEX 2

RULE soustraction: ote de
(binary_op:+asf) => "ote de"
LEX 2

RULE soustrait
(binary_op:+asf) => "soustrait a"
LEX 2

RULE retranche
(binary_op:+asf) => "retranche a"
LEX 2
```

```
RULE multiplie par
(binary_op:+mdf) => "multiplie par"
LEX 3

RULE fois
(binary_op:+mdf) => "fois"
LEX 3

RULE division: sur
(binary_op:+mdf) => "sur"
LEX 4

RULE division: pour
(binary_op:+mdf) => "pour"
LEX 4

RULE divise par
(binary_op:+mdf) => "divise par"
LEX 4

RULE en
(binary_op:+mdf) => "en"
LEX 6

RULE exponentiation: eleve a la puissance
(binary_op:+exf) => "eleve a la puissance"
LEX 5

RULE exponentiation: puissance
(binary_op:+exf) => "puissance"
LEX 5

RULE egal
(comparator:+is+than) => "egal"
LEX 0

RULE egale
(comparator:+is) => "egale"
LEX 0

RULE equivalent a
(comparator:+than) => "equivalent"
LEX 0

RULE meme que
(comparator:+than) => "la meme chose"
LEX 0
```

```
RULE plus grand que
<comparator:+than> => "plus grand"
LEX 1

RULE depasse
<comparator:+is> => "depasse"
LEX 1

RULE plus de
<comparator:+than> => "plus"
LEX 1

RULE plus eleve que
<comparator:+than> => "plus eleve"
LEX 1

RULE au dela de
<comparator:+than> => "au dela"
LEX 1

RULE moins de
<comparator:+than> => "moins"
LEX 2

RULE plus bas que
<comparator:+than> => "plus bas"
LEX 2

RULE plus petit que
<comparator:+than> => "plus petit"
LEX 2

RULE en dessous de
<comparator:+than> => "en dessous"
LEX 2

RULE plus grand ou egal a
<comparator:+than> => "plus grand ou egal"
LEX 3

RULE aussi eleve que
<comparator:+than> => "aussi eleve"
LEX 3

RULE aussi grand que
<comparator:+than> => "aussi grand"
LEX 3
```

```
RULE autant que
<comparator:+than> => "autant"
LEX 3

RULE plus petit ou egal a
<comparator:+than> => "plus petit ou egal"
LEX 4

RULE au plus
<comparator> => "au plus"
LEX 4

RULE au moins
<comparator> => "au moins"
LEX 4

RULE as little as
<comparator:+than> => "aussi petit"
LEX 4

RULE compris entre
<comparator:+between> => "entre"
LEX 6

RULE compris entre
<comparator:+between> => "compris entre"
LEX 6

RULE strictement compris entre
<comparator:+between> => "compris strictement entre"
LEX 7

RULE strictement compris entre
<comparator:+between> => "strictement entre"
LEX 7

RULE strictement compris entre
<comparator:+between> => "strictement compris entre"
LEX 7

RULE "que"
<mot:+than> => "que"
LEX 0

RULE "de"
<mot:+than> => "de"
LEX 0
```

```
RULE "qu'"
(mot:+than+vol) => "qu'"
LEX 0

RULE "d'"
(mot:+than+vol) => "d'"
LEX 0

RULE "a"
(mot:+than) => "a "
LEX 0

** nombres literaux valides epeles en toutes lettres

RULE "zero"
(literal_number:+dig1) => "zero"
LEX 0

RULE "un"
(literal_number:+dig1+art) => "un"
LEX 1

RULE "un"
(literal_number:+dig1+art+conj) => "et un"
LEX 1

RULE "un"
(literal_number:+dig1+art) => "une"
LEX 1

RULE "un"
(literal_number:+dig1+art+conj) => "et une"
LEX 1

RULE "deux"
(literal_number:+dig1) => "deux"
LEX 2

RULE "trois"
(literal_number:+dig1) => "trois"
LEX 3

RULE "quatre"
(literal_number:+dig1) => "quatre"
LEX 4

RULE "cinq"
(literal_number:+dig1) => "cinq"
LEX 5
```

```
RULE "six"
<literal_number:+dig1> => "six"
LEX 6

RULE "sept"
<literal_number:+dig1> => "sept"
LEX 7

RULE "huit"
<literal_number:+dig1> => "huit"
LEX 8

RULE "neuf"
<literal_number:+dig1> => "neuf"
LEX 9

RULE "dix"
<literal_number:+nbs> => "dix"
LEX 10

RULE "onze"
<literal_number:+nbs> => "onze"
LEX 11

RULE "douze"
<literal_number:+nbs> => "douze"
LEX 12

RULE "treize"
<literal_number:+nbs> => "treize"
LEX 13

RULE "quatorze"
<literal_number:+nbs> => "quatorze"
LEX 14

RULE "quinze"
<literal_number:+nbs> => "quinze"
LEX 15

RULE "seize"
<literal_number:+nbs> => "seize"
LEX 16

RULE "dix sept"
<literal_number:+nbs> => "dix sept"
LEX 17
```

```
RULE "dix huit"
(literal_number:+nbs) => "dix huit"
LEX 18

RULE "dix neuf"
(literal_number:+nbs) => "dix neuf"
LEX 19

RULE "vingt"
(literal_number:+dig2) => "vingt"
LEX 20

RULE "trente"
(literal_number:+dig2) => "trente"
LEX 30

RULE "quarante"
(literal_number:+dig2) => "quarante"
LEX 40

RULE "cinquante"
(literal_number:+dig2) => "cinquante"
LEX 50

RULE "soixante"
(literal_number:+dig2+nbs) => "soixante"
LEX 60

RULE "soixante et onze"
(literal_number) => "soixante et onze"
LEX 71

RULE "quatre vingt"
(literal_number:+dig2+nbs) => "quatre vingt"
LEX 80

RULE "cent"
(literal_number:+big) => "cent"
LEX 100

RULE "mille"
(literal_number:+big) => "mille"
LEX 1000

RULE "million"
(literal_number:+real+big) => "million"
*Remarque: le flag pour les nombres entiers ne va que jusqu'a 9999
LEX 10001
```

```
** constantes

RULE "pi"
(literal_number:+real) => "pi"
LEX 10002

RULE "infini"
(literal_number:+real) => "infini"
LEX 10003

RULE "inconnu"
(literal_number:+real) => "inconnu"
LEX 10004

RULE premier
(ordinal) => "premier"
LEX 1

RULE deuxieme
(ordinal) => "deuxieme"
LEX 2

RULE second
(ordinal) => "second"
LEX 2

RULE troisieme
(ordinal) => "troisieme"
LEX 3

RULE quatrieme
(ordinal) => "quatrieme"
LEX 4

RULE cinquieme
(ordinal) => "cinquieme"
LEX 5

RULE neuvieme
(ordinal) => "neuvieme"
LEX 9

RULE onzieme
(ordinal) => "onzieme"
LEX 11

RULE douzieme
(ordinal) => "douzieme"
LEX 12
```

```
RULE treizieme
<ordinal> => "treizieme"
LEX 13

RULE quatorzieme
<ordinal> => "quatorzieme"
LEX 14

RULE quinzieme
<ordinal> => "quinzieme"
LEX 15

RULE seizieme
<ordinal> => "seizieme"
LEX 16

RULE dix-neuvieme
<ordinal> => "dix neuvieme"
LEX 19

RULE trentieme
<ordinal> => "trentieme"
LEX 30

*** Regles Lexicales pour les Phatics

RULE ok
<phatic> => "ok"
LEX 0

RULE et si
<phatic> => "et si"
LEX 0


*** Regles de ponctuation

RULE ","
<pct:+comma> => ","
PRE 2 nil noun_new

RULE ";"
<pct:+semi_colon> => ";"
PRE 2 nil noun_new

RULE ":"
<pct:+colon> => ":"
PRE 2 nil noun_new
```

```
RULE "("
(pct:+left_paren) => "("
PRE 2 nil_noun_new

RULE ")"
(pct:+right_paren) => ")"
PRE 2 nil_noun_new

RULE "?"
(pct:+question_mark) => "?"
PRE 2 nil_noun_new

RULE "["
(pct:+left_bracket) -> "["
PRE 2 nil_noun_new

RULE "]"
(pct:+right_bracket) => "]"
PRE 2 nil_noun_new

RULE elimination des blancs avant un point
(pct:i+cbl) => " " (pct)
POST 2 no_opi_noun_new

RULE elimination des blancs apres un point
(pct:i) => (oct:-cbl) " "
POST 2 no_opi_noun_new
```

```
CASK/FR/NNNW.SYN    Syntaxe de Phrase Nominale pour Mots Nouveaux
                              Version 16-Dec-1983      )

Types d'object: 1    individu
                2    nombre
                3    texte
                4    image
                5    matrice
                6    agencement
                7    ...
                "    ...
                o    ...
                n    ...

Genres:        (1    meta)
                2    entite (e.g., individual, document, etc)
                3    categorie
                4    collectivite
                5    attribut
                6    relation
                7    variable
                8    unite

The creation of new vocabulary items can be initiated by two syntax forms:

    cree genre de objet type (du nom de/:) xxx
    cree (un attribut/une relation) objet type/objet type (du nom de/:) xxx
    genre de objet type :xxx
    (attribute/relation) objet type/objet type:xxx

Typical examples are:

    cree une entite d'individu du nom de Jean
    cree une entite d'individu:Jean
    entite d'individu:Jean
    cree les entites d'individus du nom de Jean, Marie et Paul
    entites d'individus:Jean, Marie et Paul
    cree l'individu:Jean              (note: in statements creating entities,
    texte:Rapport Meteo                the word "entity" may be omitted)

    cree une categorie d'individu du nom d'homme
    categorie d'individu: homme
    categorie:homme                  (note: if the object type is missing for a
                                      class or collective, then it is assumed
                                      to be individual)
    categorie de document:Dossier Factures
```

```
cree l'attribut individu/nombre: age
attribut individu/nombre: age
attribute numerique:age (note: if only one object type is given for
                         an attribute or relation, the argument is
                         assumed to be individual, the value is the given
                         object type)
relation:parent         (note: if no object type is given for an
                         attribute or relation,it is assumed to be
                         individual/individual)
relation texte/individu:mot cle
```

If the statement does not include the name of the object to be created, e.g., "Cree un individu:", individu:", or "Cree un texte." (note: a period may be used in this case), then a corresponding item is created but no entry is made in the lexicon. This new item may then be refered to using a pronoun, provided that intervening noun phrases do not mitigate its use as the proper coref. An example is:

```
"Cree un texte.
Il s'appelle Memorandum."

"individu:
Ses parents sont Jean et Marie."
```

The following syntax:

```
cree (nom de categorie) (du nom de/:) xxx
```

will create an entity whose type is the same as the given class and will make that entity a member of this class. Thus:

```
categorie:homme
cree des homme du nom de Jean et Pierre
cree un homme:Paul
```

will create an individual class "homme" and individuals "Jean", "Pierre" and "Paul"; and will put Jean, Pierre and Paul in the class homme.

The following synonyms can be used:

```
"constante" for "entite numerique"
"fichier" for "categorie de texte"
```

Lexical rules:

```
RULE "cree"
(mot_nouv:+create) => "cree "
LEX 0

RULE "creez"
(mot_nouv:+create) => "creez "
LEX 0

RULE "delete"
(mot_nouv:+delete) => "elimine"
LEX 0

RULE "delete"
(mot_nouv:+delete) => "eliminez"
LEX 0

RULE "remove"
(mot_nouv:+delete) => "supprime"
LEX 0

RULE "remove"
(mot_nouv:+delete) => "supprimez"
LEX 0

RULE "remove"
(mot_nouv:+delete) => "enleve"
LEX 0

RULE "remove"
(mot_nouv:+delete) => "enlevez"
LEX 0

RULE "du nom de"
(mot_nouv:+named) => "du nom de"
LEX 0

RULE "du nom d'"
(mot_nouv:+named+vol) => "du nom d'"
LEX 0

RULE "nomme"
(mot_nouv:+named) => "nomme"
LEX 0

RULE "objet"
(mot_nouv:+type+mas+vol) => "objet"
LEX 1
```

```
RULE "individu"
(mot_nouv:+type+mas+vol) => "individu"
LEX 1

RULE"numerique"
(mot_nouv:+type) => "numerique"
LEX 2

RULE "nombre"
(mot_nouv:+type+mas) => "nombre"
LEX 2

RULE "constante"
(mot_nouv:+type+const+fem) => "constante"
LEX 2

RULE "texte"
(mot_nouv:+type+mas) => "texte"
LEX 3

RULE "image"
(mot_nouv:+type+fem+vol) => "image"
LEX 4

RULE "matrice"
(mot_nouv:+type+fem) => "matrice"
LEX 5

RULE "agencement"
(mot_nouv:+type+mas+vol) => "agencement"
LEX 6

RULE "entite"
(mot_nouv:+kind+fem+vol) => "entite"
LEX 2

RULE "categorie"
(mot_nouv:+kind+class+fem) => "categorie"
LEX 3

RULE "collectivite"
(mot_nouv:+kind+fem) => "collectivite"
LEX 4

RULE "attribut"
(mot_nouv:+kind+mas+vol) => "attribut"
LEX 5
```

```
RULE "relation"
(mot_nouv:+kind+fem) => "relation"
LEX 6

RULE "variable"
(mot_nouv:+kind+fem) => "variable"
LEX 7

RULE "unite"
(mot_nouv:+unit+fem)=> "unite"
LEX 8


Regles de Grammaire:

RULE "create" article_defini
(mot_nouv:i+dtf) =>
     (mot_nouv:+create) (article_defini)
SYN 2 no_opi_noun_new

RULE "create" article_indefini
(mot_nouv:i+dtf) =>
     (mot_nouv:+create) (article_indefini)
SYN 2 no_opi_noun_new

RULE delete " "
(mot_nouv:i+blk) =>
     (mot_nouv:+delete-blk) " "
SYN 2 no_opi_noun_new

RULE "delete" article_defini
(mot_nouv:i+dtf) =>
     (mot_nouv:+delete+blk) (article_defini)
SYN 2 no_opi_noun_new

RULE "delete" article_indefini
(mot_nouv:i+dtf-blk) =>
     (mot_nouv:+delete+blk) (article_indefini)
SYN 2 no_opi_noun_new

RULE mot_nouv "e"
(mot_nouv:i+fem-mas) => (mot_nouv:-id-create-delete) "e"
SYN 2 no_opi_noun_new

RULE mot_nouv "s"
(mot_nouv:i+plf) => (mot_nouv:-id-create-delete) "s"
SYN 2 no_opi_noun_new
```

```
RULE mot_nouv "anime"
<mot_nouv:i+anim> => <mot_nouv:-id> " anime"
SYN 2 no_op1_noun_new

RULE ":"
<mot_nouv:+colon> => ":"
SYN 2 nil_noun_new

RULE " :"
<mot_nouv:+colon> => " :"
SYN 2 nil_noun_new

RULE id: <type>
<mot_nouv:i+id> => <mot_nouv:+type-id>
SYN 2 no_op0_noun_new

RULE id: <unit>
<mot_nouv:i+id> => <mot_nouv:+unit-id>
SYN 2 no_op0_noun_new

RULE id: <kind>
<mot_nouv:i+id> => <mot_nouv:+kind-id>
SYN 2 no_op0_noun_new

RULE id: <kind> <type>
<mot_nouv:i+id> => <mot_nouv:+kind-id> " "
                   <mot_nouv:+type-id-const>
SYN 2 no_op0_noun_new

RULE id: <kind> "de" <type>
<mot_nouv:i+id> => <mot_nouv:+kind-id> " " <preposition:+of>
                   <mot_nouv:+type-id-const>
SYN 2 no_op0_noun_new

RULE id: <kind> <type>/<type>
<mot_nouv:i+id> => <mot_nouv:+kind-id> " " <mot_nouv:+type-id-const> "/"
                   <mot_nouv:+type-id-const>
SYN 2 no_op0_noun_new

RULE "fichier" = "categorie de texte"
<mot_nouv:+id+mas> => "fichier"
SYN 2 new_word_file

RULE cree <mot_nouv:id> nomme
<sentence> => <mot_nouv:+create>
        <mot_nouv:+id> " " <mot_nouv:+named-vol> " "
SYN 2 new_word_syn
```

```
RULE cree <mot_nouv:id> nomme
<sentence> => <mot_nouv:+create>
        <mot_nouv:+id> " " <mot_nouv:+named+vol>
SYN 2 new_word_syn

RULE cree <mot_nouv:id> :
<sentence> => <mot_nouv:+create>
              <mot_nouv:+id> <mot_nouv:+colon>
SYN 2 new_word_syn

RULE cree <mot_nouv:+id> .
<sentence> <right_delimiter> => <mot_nouv:+create>
              <mot_nouv:+id> "." <right_delimiter>
SYN 2 new_word_syn

RULE cree <mot_nouv:+id>
<sentence> <right_delimiter> => <mot_nouv:+create>
              <mot_nouv:+id> <right_delimiter>
SYN 2 new_word_syn

RULE <mot_nouv:id>:
<left_delimiter> <sentence> =>
      <left_delimiter> <mot_nouv:+id>
      <mot_nouv:+colon>
SYN 2 new_word_syn

RULE cree <nom> nomme
<sentence> => <mot_nouv:+create>
        <nom:+class> " " <mot_nouv:+named-vol> " "
SYN 2 new_word_syn

RULE cree <nom> nomme
<sentence> => <mot_nouv:+create>
        <nom:+class> " " <mot_nouv:+named+vol>
SYN 2 new_word_syn

RULE cree <nom> :
<sentence> => <mot_nouv:+create>
              <nom:+class> <mot_nouv:+colon>
SYN 2 new_word_syn

RULE cree <nom> .
<sentence> <right_delimiter> => <mot_nouv:+create>
              <nom:+class> "." <right_delimiter>
SYN 2 new_word_syn
```

```
RULE cree <nom>
<sentence> <right_delimiter> => <mot_nouv:+create>
            <nom:+class> <right_delimiter>
SYN 2 new_word_syn

RULE delete <new_word:id> named
<sentence> => <mot_nouv:+delete+blk>
        <mot_nouv:+id> " " <mot_nouv:+named>
SYN 2 new_word_syn

RULE delete <new_word:id> :
<sentence> => <mot_nouv:+delete+blk>
              <mot_nouv:+id> <mot_nouv:+colon>
SYN 2 new_word_syn

RULE delete <noun>
<sentence> => <mot_nouv:+delete> <mot_nouv:+colon>
SYN 2 new_word_syn

RULE "variable"
<nom:+variable> => <double_quote> <phrase_nom>
    <double_quote>
SYN 2 quote_variable



*** [ASK/UL/NNCO.SYN]

*   Common Noun Phrase Semantics


RULE adding "al" to noun stem
<nom:i-noe> => <racine_de_nom:-noe> "al"
* e.g., "cheval"
SYN 7 no_opi_Noun_Common

RULE adding "aux" to noun stem
<nom:i+plf-noe> => <racine_de_nom:-noe> "aux"
* e.g., "chevaux"
SYN 7 no_opi_Noun_Common

RULE adding "al" to noun stem
<nom:i+noe> => <racine_de_nom:+noe> "al"
* e.g., "cheval"
SYN 7 no_opi_Noun_Common
```

```
RULE adding "aux" to noun stem
(nom:1+plf+noe) => (racine_de_nom:+noe) "aux"
* e.g., "chevaux"
SYN 7 no_opi_Noun_Common

RULE adding "e" to noun
(nom:1+fem) => (nom:-fem-noe) "e"
* e.g., "grise"
SYN 7 no_opi_Noun_Common

RULE adding "s" to noun
(nom:1+plf) => (nom:-plf) "s"
* e.g., "states"
SYN 7 no_opi_Noun_Common

RULE adding "x" to noun
(nom:1+plf) => (nom:-plf) "x"
* e.g., "hiboux"
SYN 7 no_opi_Noun_Common

RULE contract verbe aux. and following space
(verbe_aux:1) => (verbe_aux) " "
*e.g. "est-ce quela terre"  => "est-ce que la terre"
SYN 7 no_opi_Noun_Common

RULE contract article and following space
(article_indefini:1) => (article_indefini) " "
*e.g. "ungarcon"  => "un garcon"
SYN 7 no_opi_Noun_Common

RULE contract article and following space
(article_defini:1) => (article_defini) " "
*e.g. "legarcon"  => "le garcon"
SYN 7 no_opi_Noun_Common

RULE contract preposition and following space
(preposition:1) => (preposition) " "
*e.g. "Soeur deMarc"  => "Soeur de Marc"
SYN 7 no_opi_Noun_Common

RULE contract word and following space
(mot:1) => (mot) " "
*e.g. "quelui"  => "que lui"
SYN 7 no_opi_Noun_Common
```

```
*** Rules that build noun phrases from nouns

RULE individual noun to noun phrase
(phrase_nom:1-entity+class+sgf) =>
        (nom:+entity-unit)
SYN 6   ent_noun_to_noun_phrase

RULE class noun to noun phrase
(phrase_nom:1) => (nom:+class-meta)
SYN 11  class_noun_to_noun_phrase

RULE attribute noun to noun phrase
(phrase_nom:1) => (nom:+attribute)
SYN 12  attribute_noun_to_noun_phrase

***  Articles and parentheses with noun phrases


RULE putting a definite article on a noun phrase
(phrase_nom:2+dtf+the) => (article_defini)
     (phrase_nom:-psf-cbl)
* e.g., "the ship", "the (sister of John)"
*        *"(the sister) of John",
*        *"the (boy and [the] girl)"
SYN 2  article_proc
* The literal underlying "the xxx" is replaced
* by just "xxx". This is a compromise among
* several considerations in the framing of
* diagnostics.

RULE putting "un" on a noun phrase
(phrase_nom:2+mas+dtf-cjf) => (article_indefini:+mas)
              (phrase_nom:-fem-psf-cjf)
SYN 13  article_proc

RULE putting "une" on a noun phrase
(phrase_nom:2+fem+dtf-cjf) => (article_indefini:+fem)
              (phrase_nom:-mas-psf-cjf)
SYN 13  article_proc

RULE putting "des" on a noun phrase
(phrase_nom:2+plf+dtf-cjf) => (article_indefini:+plf)
              (phrase_nom:-psf-cjf)
SYN 13  article_proc

RULE "(" noun phrase ")"
(phrase_nom:2-dtf-def-rcf-asf-mdf-exf-fnf-cbl-pmf-psf-cjf)
     => (pct:+left_paren-cbl) (phrase_nom)
        (pct:+right_paren)
* This rule simply resets all of the precedence
* features. Note that it does not reset the
* quantifier feature.
SYN 8 np_op2_Noun_Common
```

```
RULE e.g. "ship class"
(phrase_nom:1) => (mot_nouv:+class-id-plf) " "
     (preposition:+of)
     (phrase_nom:+class-dtf-meta)
* "biscuit pour chien"
SYN 2 no_op1_Noun_Common

***  Rules that build noun phrases from noun
*     phrases

**    "Attribute of class" and "attribute of
*      attribute" rules

RULE genitive mod. ("pour" phrase)
(phrase_nom:1+pmf-attribute-unary_op) =>
     (phrase_nom:+attribute-pof-dtf-qnf-psf-rcf-cjf)
      " " (phrase_prep:+for-pmf)
SYN 2  attribute_of_proc
* e.g. "bonne annee pour les Bourgognes"

RULE genitive mod. ("of" phrase)
(phrase_nom:1+pmf-attribute-unary_op) =>
     (phrase_nom:+attribute-pof-dtf-qnf-psf-rcf-cjf)
      " " (phrase_prep:+of-pmf)
SYN 2  attribute_of_proc
* Central procedure for applying an attribute to
* an argument.  It instantiates the hidden
* argument of the first, attribute, constituent
* by the second constituent, then sets up a POST
* procedure, allowing the semantic processor to
* complete the evaluation.
* e.g., "sister of John"
*       "sister of whom"
*       "destination of what"
*       "sine of 3"
* Builds a complex attribute from two attributes.
* If either constituent is a relation, then so
* is the result. It instantiates the hidden
* argument of the first, attribute, constituent
* by the second, attribute constituent, leaving
* the hidden argument of the second attribute
* un-instantiated.
* e.g., "father of parent"
*       "square root of length"
*       "sum of incomes [of employees]"
*       "sine of cos","ln of sqrt"
```

```
**   Possessive rules

RULE possessive modification of an attribute
*     noun by ent. or class noun
(phrase_nom:2+psf-attribute-relation-cjf-unary_op-cjf-and-or) =>
     (phrase_nom:-attribute-pmf-rcf-dtf+pof) " "
     (phrase_nom:+attribute-pmf-rcf-dtf-qnf-psf-cjf)
* e.g., "John's sister"
*       *"John's sister and brother"
SYN 2  attribute_of_proc
* This is the central procedure for applying an
* attribute or relation to an argument. It
* instantiates the hidden argument of the
* second, attribute or relation, constituent by
* the first constituent, then sets up a POST
* procedure and calls the semantic processor.

RULE possessive modification of an attribute or
*     relation noun
*     by relative pronoun
(phrase_nom:2-attribute-relation+psf-cjf-and-or) =>
        (pronom_relatif:+whose) " "
        (phrase_nom:+attribute-dtf-qnf-pmf-psf-rcf)
* e.g., "whose sister"
SYN 2  attribute_of_proc

RULE possessive modification of an attribute or
*     relation noun by an attribute or relation
*     noun.
(phrase_nom:2+psf-cjf-and-or) =>
       (phrase_nom:+attribute-unary_op-dtf-pmf-rcf+pof) " "
       (phrase_nom:+attribute-unary_op-dtf-pmf-rcfu-qnf-psf)
* e.g., "parent's father"
SYN 2  attribute_of_proc
* Builds a complex attribute from two attributes.
* If either are relations, then the result is a
* relation. It instantiates the hidden argument
* of the second, attribute or relation,
* constituent, leaving the hidden argument of
* the first, attribute or relation, constituent
* un-instantiated.

RULE (9) number att. noun  than  noun phrase
(phrase_nom:+number+class+fnf) =>
(phrase_nom:+attribute-unary_op-pof-pmf-psf-rcf-qnf+comp-dtf-asf-mdf-exf-cjf)
     " "(mot:+than) (phrase_nom:-attribute+pof)
* e.g., "greater age than John's"
SYN 2 attribute_of_proc
```

```
**   Application of unary operators

RULE (10) unary operator ( number )
(phrase_nom:2+fnf-dtf) =>
   (phrase_nom:+number+unary_op-postfix-def-cbl-dtf-pof)
   (phrase_nom:+number-attribute-comp-asf-mdf-pbj-exf-fnf-pof-pmf)
* e.g., "sine 3", "sin(3)"
* i.e., disallow (3+sin)(pi) unless by
* definition, but allow (sin**2)(pi)
*       "average(age of students)",
*       "max(estimated times of events)"
SYN 2  attribute_of_proc

RULE (10) unary operator ( number )
(phrase_nom:2+fnf-dtf) =>
   (phrase_nom:+number+unary_op-postfix-def-cbl-dtf-pof+rsn) " "
   (phrase_nom:+number-attribute-comp-asf-mdf-pbj-exf-fnf-pof-pmf)
* e.g., "sin**3 45"
SYN 2  attribute_of_proc

*RULE (12) number   postfix unary operator, e.g., "34 squared"
*(phrase_nom:1+number+exf) =>
*     - (phrase_nom:+number-comp-cjf-asf-mdf-pbj-exf-dtf)
*       (phrase_nom:+number+unary_op+postfix-def)
* e.g., "34 squared"; parse "sin 3 squared" as
*       "(sin 3) squared"
*SYN 2  attribute_of_proc

RULE (13) unary operator  number att. noun
(phrase_nom:2-dtf+fnf-psf) =>
   (phrase_nom:+number+unary_op-postfix-def-cbl-dtf)
   (phrase_nom:+number+attribute-unary_op-pof-def-comp-cjf-asf-mdf-exf-fnf-cbl)
* e.g., "log income"
SYN 2 attribute_of_proc

RULE (13) unary operator  number att. noun
(phrase_nom:2-dtf+fnf-psf) =>
   (phrase_nom:+number+unary_op-postfix-def-cbl-dtf+rsn) " "
   (phrase_nom:+number+attribute-unary_op-pof-def-comp-cjf-asf-mdf-exf-fnf-cbl)
* e.g., "log income"
SYN 2 attribute_of_proc

* Unary operator composition:
* note that two features are important to
* propagate: cls and sgn; attribute_of_proc
* sets uop+cls if either uop arg has it
```

```
RULE number or num. att.   postfix unary operator
(phrase_nom:1+exf) =>
(phrase_nom:+number-unary_op-pof-def-cjf-comp-asf-mdf-exf-dtf-pbj)
" " (phrase_nom:+number+unary_op+postfix-def-sgn-cbl)
* e.g., "length squared"
*       "34 squared"; parse "sin 3 squared" as
*       "(sin 3) squared"
SYN 2 attribute_of_proc

*RULE (15) number att. noun   postfix unary operator
*(phrase_nom:1+exf) =>
*(phrase_nom:+number+attribute-unary_op-pof-def-cjf-comp-asf-mdf-exf-dtf)
*" " (phrase_nom:+number+unary_op+postfix-def)
* e.g., "length squared"
*SYN 2 attribute_of_proc

* condition for next two is that we can have uop+sgn at very front
* but not in the middle (then fnf is set)

RULE (16) unary operator composition: unary_op unary_op
(phrase_nom:1+fnf) =>
(phrase_nom:+number+unary_op-postfix-asf-mdf-fnf-cbl-dtf)
(phrase_nom:+number+unary_op-postfix-asf-mdf-fnf-cbl-sgn)
* e.g., "minus exp","ln(sqrt)", "square minus"
* here we start composition with right-most operator, this allows sgn to
* be on in this position
SYN 2 attribute_of_proc

RULE (17) noun_phrase:+number+attribute+unary_op composition, prefix prefix
(phrase_nom:1+fnf) =>
(phrase_nom:+number+unary_op-postfix-asf-mdf-fnf-cbl-dtf)
(phrase_nom:+number+unary_op-postfix-sgn-asf-mdf+fnf-cbl)
* e.g., "minus exp","ln(sqrt)"
* in a middle position, sgn must be off, preventing "ln-sqr"
* from parsing as ln(-(sqrt)) and "sin + cos" as "sin(+(cos))"
SYN 2 attribute_of_proc

RULE unary operator composition: unary_op unary_op
(phrase_nom:1+fnf) =>
(phrase_nom:+number+unary_op-postfix-asf-mdf-fnf-cbl-dtf+rsn) " "
(phrase_nom:+number+unary_op-postfix-asf-mdf-fnf-cbl-sgn)
* e.g., "sin**2 sqrt"
SYN 2 attribute_of_proc

RULE  noun_phrase:+number+attribute+unary_op composition, prefix prefix
(phrase_nom:1+fnf) =>
(phrase_nom:+number+unary_op-postfix-asf-mdf-fnf-cbl-dtf+rsn) " "
(phrase_nom:+number+unary_op-postfix-sgn-asf-mdf+fnf-cbl)
SYN 2 attribute_of_proc
```

```
RULE (18) unary operator composition, anything postfix
(phrase_nom:1+exf) =>
(phrase_nom:+number+unary_op-sgn-asf-mdf-fnf-psf-cbl-dtf)
(phrase_nom:+number+unary_op+postfix-sgn-asf-mdf-fnf-exf-cbl)
* e.g., "sin squared", "sin squared cosine"
* first arg can be anything but a sign
* att_of_class will invert the arguments when applying postfix operators
SYN 2 attribute_of_proc

RULE (19) uop vector of number attribute
(vecteur:+attribute+number) =>
   (vecteur:+uop-postfix-dtf) " "
   (preposition:+of)
   (phrase_nom:+number+attribute-pmf-def-pof-rcf-cjf)
* e.g., "list mean and std dev of lengths of ships" parses:
*           (                                )
* and thus gives the weighted average
SYN 2 attribute_of_proc

RULE (19) uop vector number attribute
(vecteur:+attribute+number) => (vecteur:+uop-postfix-dtf) " "
     (phrase_nom:+number+attribute-pmf-def-pof-rcf-cjf)
* e.g., "list mean and std dev of lengths of ships" parses:
*           (                                )
* and thus gives the weighted average
SYN 2 attribute_of_proc

RULE (20) uop vector of number attribute
(vecteur:+attribute+number) =>
     (phrase_nom:+number+attribute+pof-def-pmf-psf-rcf-cjf-dtf)
     (vecteur:+uop-postfix-dtf)
* e.g., "list ships' lengths' mean and std dev" parses:
*                  (                          )
* and thus gives the weighted average
SYN 2 attribute_of_proc

***  Quantifier Rules

RULE quantification of a plural noun
(phrase_nom:2+qnf-cjf) => (quantificatif:+plf-num) " "
          (phrase_nom:+plf-psf-qnf-dtf-pnf)
* e.g., "some ships"
*       "what tankers and trawlers"
POST 4  quant_proc

RULE quantification of a singular noun
(phrase_nom:2+qnf-cjf) =>
       (quantificatif:+sif-num) " "
       (phrase_nom:-plf-psf-qnf-dtf-pnf)
* e.g., "some ship"
*       "each tanker and trawler"
POST 4  quant_proc
```

```
RULE quantification of a pronoun
(phrase_nom:2+qnf-cjf) =>
          (quantificatif:+otf-num) " "
          (phrase_prep:+of)
* e.g., "some of those"
*       "some of John's dogs"
*       "some of the ships"
*       "some of the tankers and the trawlers"
POST 4  quant_proc

RULE number quantification of a plural noun
(phrase_nom:3+qnf-cjf) =>
     (quantificatif:+plf+num) " " (whole_number) " "
     (phrase_nom:-number+plf-psf-qnf-dtf-pnf)
* e.g., "at least 5 ships"
POST 4  quant_proc

RULE number quantification of a singular noun
(phrase_nom:3+qnf-cjf) =>
     (quantificatif:+sif+num) " " (whole_number) " "
     (phrase_nom:-number-plf-psf-qnf-dtf-pnf)
* e.g., "exactly 1 ship"
POST 4  quant_proc

RULE number quantification of a pronoun
(phrase_nom:3+qnf-cjf) =>
     (quantificatif:+otf+num) " " (whole_number) " "
     (phrase_prep:+of-number)
* e.g., "at least 5 of those"
*       "at least 5 of John's dogs"
*       "at least 5 of the ships"
POST 4  quant_proc

RULE whole number as quantifier
(phrase_nom:+qnf-cjf) => (whole_number:-art) " "
     (phrase_nom:-number-plf-psf-qnf-dtf-pnf)
* e.g. "20 ships"
* note: this is interpreted as "exactly 20"
* thus,
*  >Are there 20 ship?
*   exactly 20 ships
*   no
POST 2 num_quant_proc
```

```
***  Rules for forming vectors

RULE "the" attribute vector
(vecteur:2+dtf) => (article_defini)
     (vecteur:-dtf)
SYN 13 article_proc

RULE remove parens from vector
(vecteur:2-cjf-dtf) => (pct:+left_paren-cbl)
     (vecteur) (pct:+right_paren)
SYN 8 no_op2_Noun_Common


*** Unit rules

*RULE plural units "s"
*(phrase_nom:1+plf) =>
*(phrase_nom:+number+unit-cls-comp-cjf-asf-mdf-pbj-exf-fnf-plf) "s"
*SYN 19 num_article_proc
```

```
(ASK/FR/NNVE.SYN )

(      Noun Phrase Syntax for Vector Attribute Phrases      )

* form attribute vectors

RULE attribute conj attribute
(vecteur:+attribute) =>
    (phrase_nom:+attribute-number-cjf-dtf)
    (conj:+and) (phrase_nom:+attribute-cjf)
SYN 2  att_vector_conj

RULE attribute conj attribute
(vecteur:+attribute) =>
    (phrase_nom:+attribute+number-cjf-dtf)
    (conj:+and)
    (phrase_nom:+attribute-number-cjf)
SYN 2  att_vector_conj

RULE attribute conj attribute
(vecteur:+attribute+number) =>
    (phrase_nom:+attribute+number-cjf-dtf)
    (conj:+and)
    (phrase_nom:+attribute+number-cjf)
SYN 2  att_vector_conj

RULE attribute conj attribute
(vecteur:+attribute+comma) =>
    (phrase_nom:+attribute-number-cjf-dtf)
    (conj:+comma) (phrase_nom:+attribute-cjf)
SYN 2  att_vector_conj

RULE attribute conj attribute
(vecteur:+attribute+comma) =>
    (phrase_nom:+attribute+number-cjf-dtf)
    (conj:+comma)
    (phrase_nom:+attribute-number-cjf)
SYN 2  att_vector_conj

RULE attribute conj attribute
(vecteur:+attribute+number+comma) =>
    (phrase_nom:+attribute+number-cjf-dtf)
    (conj:+comma)
    (phrase_nom:+attribute+number-cjf)
SYN 2  att_vector_conj
```

```
RULE attribute conj vector
(vecteur:3-number+and) =>
   (phrase_nom:+attribute-number-cjf-dtf)
   (conj:+and)
   (vecteur:+attribute-comma)
SYN 2  att_vector_conj

RULE vector conj attribute
(vecteur:3+and) =>
   (phrase_nom:+attribute+number-cjf-dtf)
   (conj:+and)
   (vecteur:+attribute-comma)
SYN 2  att_vector_conj

RULE attribute conj vector
(vecteur:3+comma) =>
   (phrase_nom:+attribute+number-cjf-dtf)
   (conj:+comma)
   (vecteur:+attribute-and)
SYN 2  att_vector_conj

RULE attribute "," vector
(vecteur:3-number+comma) =>
   (phrase_nom:+attribute-number-cjf-dtf)
   (conj:+comma)
   (vecteur:+attribute-and)
SYN 2  att_vector_conj

RULE attribute vector "versus"
(vecteur:+attribute+number) =>
   (phrase_nom:+attribute+number-cjf-dtf)
   " " (mot:+versus) " "
   (phrase_nom:+attribute+number-cjf)
SYN 2  att_vector_conj

* now apply attribute vectors to noun phrases

RULE vector of argument
(vecteur:1-attribute) => (vecteur:+attribute-dtf) " "
   (phrase_prep:+of)
PRE 3  att_vector_of

RULE argument's vector
(vecteur:2-attribute) => (phrase_nom:+pof-dtf) " "
   (vecteur:+attribute)
PRE 3  att_vector_of
```

```
(ASK/FR/NNAD.SYN)
(       Noun Phrase Semantics -        )

RULE genitive modification of an ind. or class noun by an ind. or class noun
(phrase_nom:i+pmf-cjf) =>
    (phrase_nom:-pof-rcf-dtf-qnf-psf-attribute)
    " " (phrase_prep:+of-pmf)
* e.g., "cities of China"
POST 3  adj_noun_proc
***  For french, the semantics are the same
***  as the next rule, because "vin d'Alsace"
***  can also have the elliptical sense of "Alsace".


**    Nouns in an adjectival relationship

RULE adjectival modification of a class noun
(phrase_nom:i+apf-cjf) =>
      (phrase_nom:-dtf-qnf-pmf-psf-rcf-attribute-number) " "
      (phrase_nom:-pof-dtf-qnf-apf-pmf-psf-rcf-attribute-number-meta)
* e.g., "male dog", "Boston ships", "Boston individuals"
POST 2  adj_noun_proc
* Intersects the two constituents; if the intersection is empty, it looks
* for an intervening attribute from 2nd constituent to 1st constituent
* or from 1st to 2nd.

RULE adjectival modification of an attribute noun
(phrase_nom:i+apf-cjf) =>
      (phrase_nom:-dtf-qnf-pmf-psf-rcf+attribute-number) " "
      (phrase_nom:-pof-dtf-qnf-apf-pmf-psf-rcf-number-meta)
* e.g., "male parent"
POST 4  adj_noun_proc
* Result is a complex attribute that intersects the 1st constituent with
* the result of applying the 2nd, attribute constituent to an argument.
* If the 1st constituent is an attribute, then it is replaced by its range.

RULE adjectival modification of a noun
(phrase_nom:i+pmf-cjf) =>
(phrase_nom:-pof-qnf-psf-rcf-number-cjf)
      " " (phrase_prep:+in-pmf)
* e.g., "ships in Task Force 53"
POST 4  adj_noun_proc

RULE adjectival modification by an attribute noun
(phrase_nom:i+apf-cjf) =>
(phrase_nom:-dtf-qnf-pmf-psf-rcf-attribute-number) " "
(phrase_nom:-pof-dtf-qnf-apf-pmf-psf-rcf+attribute-number)
* e.g., "author Scott", "port city"
POST 4  class_intersection_proc
* Intersects the two constituents; note that the attribute will already
* be reduced to its range, thus the result is the class of all those members
* of the second constituent that are in the range of the attribute.
```

```
(ASK/FR/NNRC.SYN  )

(  Noun Phrase Semantics for Relative Clauses )

***  Relative clauses

**   Rules that build relative clauses

RULE building a relative clause with "who," "which," or "that"
(prop_relative) => (pronom_relatif:-whom-whose-what) " " (phrase_verb:-fqt)
* e.g., "[the commander] who issued the report",
*       "[the commander] who the captain told",
*       "[the report] that the commander issued ",
*       "[the report] that was sent to the captain",
*       "[the report] which the commander issued ",
*       "[the report] which was sent to the captain"
PRE 10  no_op2_Noun_Rel_Clause

RULE building a relative clause with "whom"
(prop_relative:+obj) => (pronom_relatif:+whom) " "
       (phrase_verb:-fqt-fpa-foj-fpp)
* e.g., "[the commander] whom the captain saw"
PRE 10  no_op2_Noun_Rel_Clause

RULE building a relative clause with "to whom/which"
(prop_relative:+dat) => "a " (pronom_relatif:-who-what-that-whose) " "
       (phrase_verb:-fqt-fda-fpp)
* e.g., "[the commander] to whom (the report was issued)",
*       "[the report] to which (the commander refered)"
PRE 10  no_op2_Noun_Rel_Clause

RULE building a relative clause with "whom/which ... to"
(prop_relative:+dat) => "a " (pronom_relatif:-whom-what-whose) " "
       (phrase_verb:-fqt-fda+fpa)
* e.g., "[the commander] who (the report was given) to",
*       "[the report] which (the commander was refered) to"
PRE 10  no_op2_Noun_Rel_Clause

RULE building a relative clause with "by whom"
(prop_relative:+sbj) => "par " (pronom_relatif:-who-that-what-whose) " "
       (phrase_verb:+fpa-fqt-fda-fpp)
* e.g., "[the commander] by whom (the report was issued)"
PRE 10  no_op2_Noun_Rel_Clause

RULE building a relative clause with "whose"
(prop_relative:+whose) => (pronom_relatif:+whose) " "
       (phrase_nom:-prep-pof-dtf-qnf) " " (phrase_verb:-fin-fqt-fsj-fpp-fcj)
* e.g., "[the commander] whose report (was issued)",
*       "[the report] whose author (gave it to the commander)"
PRE 3  whose_pro
```

```
**   Building a relative clause by a "with" prepositional phrase

RULE prepositional "with" phrase modification
(prop_relative:+with) => (phrase_nom:+prep-pof)
* e.g., "[ships] with (a doctor)",
*       "[ships] with (Soviet flag)",
*       "[ships] with (each classification)",
*       "[ships] with (at least 5 cargo spaces)"
PRE 10 no_op1_Noun_Rel_Clause

RULE prepositional "with" phrase modification, value "as only" attribute
(prop_relative:+with) => (phrase_nom:+prep-pof)
     " comme " (adverbe:+only) " "
     (phrase_nom:-prep+attribute-qnf-pof)
* e.g., "[ships] with London as only destination",
*       "[ships] with (some Soviet port) as only destination"
PRE 10 no_op2_Noun_Rel_Clause

RULE prepositional "with" phrase modification, value "only ... as" attribute
(prop_relative:+with) => (preposition:+with)
         (adverbe:+only) " " (phrase_nom:-pof)
         " comme " (phrase_nom:+attribute-qnf-pof)
* e.g., "[ships] with only trucks as cargo",
*       "[ships] with only (some Soviet port) as destination"
PRE 10 no_op2_Noun_Rel_Clause

RULE prepositional "with" phrase modification, whole number - attribute
(prop_relative:+with) => (preposition:+with)
         (whole_number) " " (phrase_nom:-pof)
* e.g., "[ships] with 20 (cargo spaces)"
PRE 10 no_op2_Noun_Rel_Clause

RULE prepositional "with" phrase modification, number - number attribute
(prop_relative:+with) => (phrase_nom:+prep+number) " "
         (phrase_nom:-prep+number+attribute-pof)
* e.g., "[cargo] with (a 750.5 foot) length"
PRE 10 no_op2_Noun_Rel_Clause

RULE prepositional "with" phrase modification, number "as" number att.
(prop_relative:+with) => (phrase_nom:+prep+number) " as "
         (phrase_nom:-prep+number+attribute-pof)
* e.g., "[cargo] with (750.5 feet) as length"
PRE 10 no_op2_Noun_Rel_Clause
```

```
RULE prepositional "with" phrase modification, number att., comparator, num.
(prop_relative:+with) => (phrase_nom:+prep+attribute-pof) " "
        (phrase_nom:-prep-pof)
* e.g., "[ships] with length (greater than 760)"
PRE 10 no_op2_Noun_Rel_Clause

RULE prepositional "with" phrase modification, num. att., coomp., poss. noun
(prop_relative:+with) => (phrase_nom:+prep+number+attribute-pof)
        " " (comparator) " " (phrase_nom:-prep-number+pof)
* e.g., "[ships] with length greater than (the Alamo's)"
PRE 10 no_op2_Noun_Rel_Clause

** modifying a noun by a relative clause

RULE modifying a noun by a relative clause
(phrase_nom:1+rcf-cjf) => (phrase_nom:-pof-dtf-qnf-rcf) " "
        (prop_relative)
* e.g., "(the commander) (who issued the report)"
PRE 2  rel_clause_proc

RULE modifying a noun by a relative clause, "," case
(phrase_nom:1+rcf-cjf) => (phrase_nom:-pof-dtf-qnf-rcf) ", "
        (prop_relative)
* e.g., "(the commander) (who issued the report)",
*       "the author of the report, who sent it to the commander"
PRE 2  rel_clause_proc

RULE modifying a noun by a relative clause, "," ... "," case
(phrase_nom:1+rcf-cjf) => (phrase_nom:-pof-dtf-qnf-rcf) ", "
        (prop_relative) ","
* e.g., "(the commander) (who issued the report)"
*       "Capt Spruance, who issued the report, ..."
PRE 2  rel_clause_proc
```

```
CASK/FR/NNMI.SYN          Noun Phrase Syntax          )


RULE name of
(nom:+attribute) => (mot:+name)
* e.g., "name of ...", "member of ...", "item of ..."
PRE 2  names_of_proc

RULE number of
(phrase_nom:+number) => (mot:+num_of) " "
                          (phrase_nom:-prep-pof)
POST 2  num_of_proc

RULE count
(sentence) => (mot:+count) " " (phrase_nom:-prep-pof) "."
POST 2  num_of_proc

RULE converse
(phrase_nom:2+pmf) => (mot:+converse) " "
          (phrase_nom:-prep+attribute-pof-pmf)
* Clearly the phrase: "converse of A1 of A2" parses as:
*     "(converse of A1) of A2", however it is construed at the actual time of
* application of the semantics as: "converse of (A1 of A2)"
PRE 2  converse_of_proc
```

```
(ASK/FR/PREP.SYN   -   Rule that builds prepositional phrases)

(Note: prep_phrase must have the same initial features since
features are inherited by prep_phrase from preposition)

(Lexical rules)

RULE "de"
(preposition:+of) => "de"
LEX 1

RULE "par"
(preposition:+by) => "par"
LEX 2

RULE "avec"
(preposition:+with) => "avec"
LEX 3

RULE "ayant"
(preposition:+with) => "ayant"
LEX 3

RULE "vers"
(preposition:+to) => "vers"
LEX 4

RULE "a"
(preposition:+to) => "a "
LEX 4

RULE "dans"
(preposition:+in) => "dans"
LEX 6

RULE "en"
(preposition:+in) => "en"
LEX 6

RULE "chez"
(preposition:+at) => "chez"
LEX 7

RULE "a propos de"
(preposition:+about) => "a propos de"
LEX 8

RULE "sur"
(preposition:+on) => "sur"
LEX 9
```

```
RULE "en temp que"
(preposition:+as) => "en temp que"
LEX 10

RULE "comme"
(preposition:+as) => "comme"
LEX 10

RULE "pour"
(preposition:+for) => "pour"
LEX 11

RULE "du"
(preposition:+of) => "du"
LEX 1

RULE "des"
(preposition:+of+olf) => "des"
LEX 1

RULE "d'"
(preposition:+of+vol) => "d'"
LEX 1

RULE "en temp qu'"
(preposition:+as+vol) => "en temp qu'"
LEX 10

(Grammar rule)

RULE prepositional phrase
(phrase_prep:1) => (preposition)
                   (phrase_nom:-pmf)
POST 2 prep_proc

RULE prepositional phrase
(phrase_prep:1+pmf) => (preposition)
                   (phrase_nom:+pmf)
POST 2 prep_proc

RULE prepositional phrase with relative pronoun
(phrase_prep:1) => (preposition)
                   (pronom_relatif:-that-which-whose)
POST 2 prep_proc
```

```
*** (ASK/FR/CONJ.SYN)


***  Basic conjunction rules

RULE "et"
(conj:+and) => "et"
LEX 1

RULE "ou"
(conj:+or) => "ou"
LEX 2

RULE "," as conj
(conj:+comma) => (pct:+comma)
SYN 2 no_op1_Conj

RULE conj with ","
(conj:2+comma) => (pct:+comma) (conj:-comma-rtsp)
SYN 2 no_op2_Conj

RULE (conj) " "
(conj:1+rtsp) => (conj:-comma) " "
SYN 2 no_op1_Conj

RULE " " (conj)
(conj:1) => " " (conj:-rtsp-comma)
SYN 2 no_op1_Conj

RULE conjunction of noun phrases, "and"
(phrase_nom:3-attribute+class-entity+cjf+and-sgf-dtf) =>
        (phrase_nom:-attribute-cjf-dtf)
        (conj:+and) (phrase_nom:-cjf)
SYN 18  conj_syn_proc

RULE conjunction of noun phrases, "and"
(phrase_nom:3+class-entity+cjf+and-sgf-dtf) =>
        (phrase_nom:+attribute-cjf-dtf)
        (conj:+and) (phrase_nom:-attribute-cjf)
SYN 18  conj_syn_proc

RULE conjunction of noun phrases, "and"
(phrase_nom:3+cjf+and-sgf-dtf) =>
        (phrase_nom:+attribute-cjf-dtf)
        (conj:+and)
        (phrase_nom:+attribute-cjf)
SYN 18  conj_syn_proc
```

```
RULE conjunction of noun phrases, "or"
(phrase_nom:3-attribute+class-entity+cjf+or-sgf+plf-dtf) =>
     (phrase_nom:-attribute-cjf-dtf) (conj:+or)
     (phrase_nom:-cjf)
SYN 18  conj_syn_proc

RULE conjunction of noun phrases, "or"
(phrase_nom:3+class-entity+cjf+or-sgf+plf-dtf) =>
     (phrase_nom:+attribute-cjf-dtf) (conj:+or)
     (phrase_nom:-attribute-cjf)
SYN 18  conj_syn_proc

RULE conjunction of noun phrases, "or"
(phrase_nom:3+cjf+or-sgf+plf-dtf) =>
     (phrase_nom:+attribute-cjf-dtf) (conj:+or)
     (phrase_nom:+attribute-cjf)
SYN 18  conj_syn_proc

RULE conjunction of noun phrases, "and" conj
(phrase_nom:3-attribute+class-dtf) =>
        (phrase_nom:-attribute-cjf-dtf)
        (conj:+and) (phrase_nom:+cjf+and)
SYN 18  conj_syn_proc

RULE conjunction of noun phrases, "and" conj.
(phrase_nom:3+class-dtf) => (phrase_nom:+attribute-cjf-dtf)
        (conj:+and) (phrase_nom:-attribute+cjf+and)
SYN 18  conj_syn_proc

RULE conjunction of noun phrases, "and" conj
(phrase_nom:3-dtf) => (phrase_nom:+attribute-cjf-dtf)
        (conj:+and) (phrase_nom:+attribute+cjf+and)
SYN 18  conj_syn_proc

RULE conjunction of noun phrases, "or" conj
(phrase_nom:3-attribute+class-dtf) =>
        (phrase_nom:-attribute-cjf-dtf)
        (conj:+or) (phrase_nom:+cjf+or)
SYN 18  conj_syn_proc

RULE conjunction of noun phrases, "or" conj
(phrase_nom:3+class-dtf) => (phrase_nom:+attribute-cjf-dtf)
        (conj:+or) (phrase_nom:-attribute+cjf+or)
SYN 18  conj_syn_proc

RULE conjunction of noun phrases, "or" conj
(phrase_nom:3-dtf) => (phrase_nom:+attribute-cjf-dtf)
        (conj:+or) (phrase_nom:+attribute+cjf+or)
SYN 18  conj_syn_proc
```

```
RULE conjunction of noun phrases, "," conj
(phrase_nom:3-attribute+class-dtf) =>
       (phrase_nom:-attribute-cjf-dtf)
       (conj:+comma-and-or) (phrase_nom:+cjf)
SYN 18  conj_syn_proc

RULE conjunction of noun phrases, "," conj
(phrase_nom:3+class-dtf) => (phrase_nom:+attribute-cjf-dtf)
       (conj:+comma-and-or) (phrase_nom:-attribute+cjf)
SYN 18  conj_syn_proc

RULE conjunction of noun phrases, "," conj
(phrase_nom:3-dtf) => (phrase_nom:+attribute-cjf-dtf)
       (conj:+comma-and-or) (phrase_nom:+attribute+cjf)
SYN 18  conj_syn_proc

RULE conjunction of verb phrases, "and"
(phrase_verb:3+fcj+and) => (phrase_verb:-fcj)
                (conj:+and) (phrase_verb:-fcj)
SYN 18  conj_syn_proc

RULE conjunction of verb phrases, "or"
(phrase_verb:3+fcj+or) => (phrase_verb:-fcj)
                (conj:+or) (phrase_verb:-fcj)
SYN 18  conj_syn_proc

RULE conjunction of verb phrases, ","
(phrase_verb:3) => (phrase_verb:-fcj)
                (conj:+comma) (phrase_verb:+fcj)
SYN 18  conj_syn_proc

RULE conjunction of relative clause phrases,
*     "and"
(prop_relative:+fcj+and) =>
               (prop_relative:-fcj) (conj:+and)
               (prop_relative:-fcj)
* e.g., "who wrote the report and who gave it
*         to the commander"
* e.g., "who wrote the report, who gave it to
*         the commander and
*         who told the captain"
SYN 18  conj_syn_proc

RULE conjunction of relative clause phrases, "or"
(prop_relative:+fcj+or) =>
               (prop_relative:-fcj) (conj:+or)
               (prop_relative:-fcj)
SYN 18  conj_syn_proc

RULE conjunction of relative clause phrases, ","
(prop_relative:3) => (prop_relative:-fcj)
            (conj:+comma) (prop_relative:+fcj)
SYN 18  conj_syn_proc
```

```
RULE process verb definition - syntax part
<mot_nouv:+verb_def> => <mot_nouv:+init+verb_def>
SYN 6  verb_syn_proc

RULE process verb definition - ":"
<mot_nouv:+verb_def+colon> => <mot_nouv:-init+verb_def-colon> ":"
SYN 5  no_opi_Verb_Def

RULE process verb definition - ": "
<mot_nouv:+verb_def+colon> => <mot_nouv:-init+verb_def-colon> ": "
SYN 5  no_opi_Verb_Def

RULE process verb definition - semantic part
<sentence> => <mot_nouv:+verb_def+colon> <phrase_verb>
PRE 8  verb_sem_proc

RULE no_opi_Verb_Def
<phrase_verb> => <verbe>
SYN 9 make_verb_phrase

RULE verb_morphology_s
<phrase_verb:+fsi> => <verbe> "s"
SYN 9 make_verb_phrase

RULE verb_morphology_es
<phrase_verb:+fsi> => <verbe> "es"
SYN 9 make_verb_phrase

RULE verb_morphology_d
<phrase_verb:+fpp> => <verbe> "e"
SYN 9 make_verb_phrase

RULE verb_morphology_ed
<phrase_verb:+fpp> => <verbe> "ent"
SYN 9 make_verb_phrase
```

```
(ASK/FR/VRBCO.SYN  )
(         The Syntax of ASK French Verb Rules  )

***  Copula Verb Rules

RULE copula to verb_phrase
<phrase_verb:+copula> => <copule>
SYN 4  copula_development_proc
* Builds a typical verb phrase, inserting the "is" or "has" proc and
* the time phrase; carries over features.

RULE comparators as copulas, e.g., "exceeds"
<phrase_verb:+foj+fph> => <comparator:+is-sym>
        " " <phrase_nom:+number-comp>
* e.g., "[5] exceeds 3"
SYN 2  is_comp_proc

RULE symbol comparators as copulas
<phrase_verb:+fsj+fag+foj+fph> =>
        <phrase_nom:+number-comp>
        <comparator:+is+sym>
        <phrase_nom:+number-comp>
* e.g., "5<=7"
SYN 2  is_comp_proc


**   Copula verbs on copula verbs as auxiliaries

RULE "have" as auxiliary to copula verb
<phrase_verb:2-fcj+fph-fpp> => <phrase_verb:+copula+have-fph> " "
        <phrase_verb:+copula+fpp-fsj-fag-foj-fce-fpa-fvi>
* e.g., "has been", "have had"
SYN 3  no_op2_Verb_Common


**   Putting predicate nouns on copula verbs

RULE agent noun in the question inversion position
<phrase_verb:i-fcj+fsj+fqt+fph+fag+fce> =>
        <phrase_verb:+copula+is-fin-fsj-foj-fag-fda-fce>
        " " <phrase_nom:-pof>
* e.g., "is John [an employ?]", "was John [a commander?]"
SYN 6  ag_proc

RULE agent RP in the question inversion position
<phrase_verb:i-fcj+fsj+fqt+fph+fag+fce> =>
        <phrase_verb:+copula+is-fin-fsj-foj-fag-fda-fce>
        " " <pronom_relatif:-that-which-whose>
* e.g., "is who [an employ?]", "was who [a commander?]"
SYN 6  ag_proc
```

```
RULE predicate noun as direct object of copula
(phrase_verb:i-fcj+fph+foj) =>
        (phrase_verb:+copula+is-fin-fvi-foj-fda)
        " " (phrase_nom:-pof)
* e.g., "(is) (a ship)", "(is Alamo) (a tanker)," "(had been) (the mayor)"
* want (vp2:+fsj+fqt+fag+fce) for the second example above
SYN 8  oj_proc

RULE predicate RP as direct object of copula
(phrase_verb:i-fcj+fph+foj+fqt) =>
        (phrase_verb:+copula+is-fin-fvi-foj-fda)
        " " (pronom_relatif:-that-which-whose)
* e.g., "(is) (what)", "(is Alamo/what) (a tanker/what)," "(had been) (what)"
* want (vp2:+fsj+fqt+fag+fce) for the second example above
SYN 8  oj_proc

RULE predicate noun as to_noun of copula
(phrase_verb:i-fcj+fsj+fqt+fph+fda+fce) =>
        (phrase_verb:+copula+is-fsj-foj-fag-fda-fce)
        " " (phrase_nom:-pof)
* e.g., "(was) (John) [sent flowers]"
SYN 8  prep_to_proc

RULE predicate RP as to_noun of copula
(phrase_verb:i-fcj+fsj+fqt+fph+fda+fce) =)
        (phrase_verb:+copula+is-fsj-foj-fag-fda-fce)
        " " (pronom_relatif:-that-which-whose)
* e.g., "(was) (who) [sent flowers]"
SYN 8  prep_to_proc

RULE "in NP" as predicate of "is"
(phrase_verb:i-fcj+foj+fph) => (phrase_verb:+copula+is-fsj)
        " " (preposition:+in) " " (phrase_nom:-pof-unit)
* e.g., "Maru is in task force 31"
SYN 2 in_proc


****  Adding case nouns to non-copula verbs


RULE direct object in normal position
(phrase_verb:i+foj+fph) =>
        (phrase_verb:-copula-fvi-fsj-foj-fag-fda-fce)
        " " (phrase_nom:-pof)
* e.g., "(carry) (coal)", "(was given) (food)"
SYN 8  oj_proc
```

```
RULE adding a RP as object of a verb, in normal position
<phrase_verb:i-fcj+foj+fph+fqt> =>
        <phrase_verb:-copula-fvi-fsj-foj-faq-fda-fce> " "
        <pronom_relatif:-that-which-whose>
* e.g., "[Who] will carry what",
SYN 14  oj_proc

RULE to_noun followed by direct object noun, without the preposition
<phrase_verb:i+foj+fda+fce+fph> =>
        <phrase_verb:-copula-fvi-fsj-faq-foj-fda-fce-fpa> " "
        <phrase_nom:-pof> " " <phrase_nom:-pof>
* e.g., "(told) (the commander) (the reason)"
SYN 10  daoj_proc


RULE RP to_noun followed by direct object noun, without the preposition
<phrase_verb:i+foj+fda+fce+fph+fqt> =>
        <phrase_verb:-copula-fvi-fsj-faq-foj-fda-fce-fpa> " "
        <pronom_relatif:-that-which-whose> " " <phrase_nom:-pof>
* e.g., "(told) (what) (the reason)"
SYN 10  daoj_proc

RULE to_noun followed by direct object RP, without the preposition
<phrase_verb:i+foj+fda+fce+fph+fqt> =>
        <phrase_verb:-copula-fvi-fsj-faq-foj-fda-fce-fpa> " "
        <phrase_nom:-pof> " " <pronom_relatif:-that-which-whose)
* e.g., "(told) (the commander) (what)"
SYN 10  daoj_proc

RULE RP to_noun followed by direct object RP, without the preposition
<phrase_verb:i+foj+fda+fce+fph+fqt> =>
        <phrase_verb:-copula-fvi-fsj-faq-foj-fda-fce-fpa> " "
        <pronom_relatif:-that-which-whose>
        " " <pronom_relatif:-that-which-whose>
* e.g., "(told) (what) (what)"
SYN 10  daoj_proc


*** Predicate: <phrase_verb> <phrase_prep>

RULE prep_noun introduced in predicate  (excluding to_noun)
<phrase_verb:i+fph+fce> => <phrase_verb:-copula-fsj-fin> " "
        <phrase_prep:-to>
* works for +fpa or -fpa
SYN 7   prep_phrase_proc
```

```
RULE to_noun introduced in predicate
(phrase_verb:1+fph+fce) => (phrase_verb:-copula-fsj-fin-fda) " "
        (phrase_prep:+to)
* works for +fpa or -fpa
SYN 7  prep_phrase_proc

****  Case Nouns in Subject Position

RULE agent noun in the subject position
(phrase_verb:2+fph+fsj+fin+fag+fce) => (phrase_nom:-pof) " "
        (phrase_verb:-fin-fpa-fsj-fag)
* e.g., "(The commander) (will tell the reason),"
*       "(The commander) (told the reason)"
SYN 6  ag_proc

RULE adding a RP as agent of a verb
(phrase_verb:2-fcj+fph+fsj+fin+fag+fce+fqt) =>
        (pronom_relatif:-that-which-whose) " "
        (phrase_verb:-fin-fpa-fsj-fag)
* e.g., "(Who/What) (tells the commander)",
*       "(Who) (is the commander)",
*       "(Who) (has the commander's report)"
SYN 12  ag_proc

RULE object noun in the subject position due to passive inversion
(phrase_verb:2+fph+fsj+fin+foj) => (phrase_nom:-pof) " "
        (phrase_verb:-copula-fin+fpa-fvi-fsj-foj)
* e.g., "(coal) (was carried by the ship)"
SYN 8  oj_proc

RULE adding a RP as object of a passive verb
(phrase_verb:2-fcj+fph+fsj+fin+foj+fqt) =>
        (pronom_relatif:-that-which-whose) " "
        (phrase_verb:-copula-fin+fpa-fvi-fsj-foj)
* e.g., "(What) (was carried)"?
SYN 14  oj_proc

RULE object noun in subject position due to wh_inversion
(phrase_verb:2+foj+fin) => (phrase_nom:+qnf-pof) " "
        (phrase_verb:-copula-fin+fsj-foj-fpa+fqt-fvi)
* e.g., "(what ship) (did Capt Jones command)",
*       "(how many trucks) (does the Alamo carry)"
SYN 8 oj_proc
```

```
RULE adding a RP as object of a verb after question inversion
(phrase_verb:2-fcj+fph+fin+foj+fqt) =>
        (pronom_relatif:-that-which-whose) " "
        (phrase_verb:-copula-fin+fsj-foj-fpa+fqt-fvi)
* e.g., "(What) (does the ship carry)",
* This does not include  "What does the Alamo have"
SYN 14  oj_proc

RULE to_noun in the subject position due to passive inversion
* and existance of an object
(phrase_verb:2+fph+fsj+fin+fda) => (phrase_nom:-pof) " "
        (phrase_verb:-copula-fin-fsj+foj-fda+fpa-fvi)
* e.g., "(Capt Jones) (was sent the report)"
SYN 9  prep_to_proc

RULE adding a RP as to_noun of a passive verb
(phrase_verb:2-fcj+fph+fsj+fin+fda+fqt) =>
        (pronom_relatif:-that-which-whose) " "
        (phrase_verb:-copula-fin-fsj+foj-fda+fpa-fvi)
* e.g., "(What) (was given an overhaul)"?
SYN 15  prep_to_proc

RULE Copula_OJ_VP for passive
(phrase_verb:2+foj+fsj+fqt+fph+fpa-fpp+test) =>
        (phrase_verb:+copula+is-fvi-fin-fsj-fcj+fph+foj-fag-fda-test)
        " " (phrase_verb:+fpp-copula-fin-fag-foj-fda-fce-test)
*e.g., "(was wheat) (carried)"
* (vp2:-fag-fda-fce) forces addition of terminal prep_phrases only
*  after the passive is formed
SYN 4 copula_vp

RULE Copula_DA_VP for passive
(phrase_verb:2+fda+fsj+fqt+fph+fpa-fpp+test) =>
        (phrase_verb:+copula+is+fqt-fin+fsj-fcj+fph-foj-fag+fda-test)
        " " (phrase_verb:+fpp-copula-fin-fag+foj-fda-fce-test)
*e.g., "(was Mary) (sent flowers)"
* (vp2:-fag-fda-fce) forces addition of terminal prep_phrases only
* after the passive is formed
SYN 4 copula_vp

RULE adding cases to Copula_VP for passive    (excluding to_noun)
(phrase_verb:1) =>
        (phrase_verb:+fsj+fqt+fph+fpa+test-copula+foj-fin-fpp)
        " " (phrase_prep:-to)
*e.g., "(was wheat carried) (by rig)"
SYN 4 prep_phrase_proc
```

```
RULE adding to_noun to Copula_VP for passive
(phrase_verb:1) =>
        (phrase_verb:+fsj+fqt+fph+fpa+test-copula+foj-fin-fpp-fda)
        " " (phrase_prep:+to)
*e.g., "(was wheat carried) (by rig)"
SYN 4 prep_phrase_proc

***(also need rules above for corresponding non-passive, +fag, +prog)


***  Initial/Terminal Preposition: (phrase_prep) (phrase_verb)  /
**                              (phrase_nom) (phrase_verb) (preposition)

RULE initial preposition on to_noun
(phrase_verb:2+fce-test) => (phrase_prep:+qnf+to)
        " " (phrase_verb:-copula+fqt+fsj-fin-fda)
* for +fpa, -fpa, +fpp, -fpp
* the vp in the right may or may not be passive.
* (copula_vp or does_vp constructs)
SYN 97 prep_phrase_proc

RULE initial preposition on prep_noun   (excluding to_noun)
(phrase_verb:2+fce-test) => (phrase_prep:+qnf-to)
        " " (phrase_verb:-copula+fqt+fsj-fin)
* for +fpa, -fpa, +fpp, -fpp
* the vp in the right may or may not be passive:
* (copula_vp or does_vp constructs)
SYN 97 prep_phrase_proc

RULE terminal preposition with inverted to_noun
(phrase_verb:2+fce-test) => (phrase_nom:+qnf-pof)
        " " (phrase_verb:-copula+fqt+fsj-fin-fda)
        " " (preposition:+to)
* for +fpa, -fpa, +fpp, -fpp
* the vp in the right may or may not be passive:
* (copula_vp or does_vp constructs)
SYN 97 inverted_prep_proc

RULE terminal preposition with inverted prep_noun    (excluding to_noun)
(phrase_verb:2+fce-test) => (phrase_nom:+qnf-pof)
        " " (phrase_verb:-copula+fqt+fsj-fin)
        " " (preposition:-to)
* for +fpa, -fpa, +fpp, -fpp
* the vp in the right may or may not be passive.
* (copula_vp or does_vp constructs)
SYN 97 inverted_prep_proc
```

```
RULE terminal preposition with inverted RP to_noun
(phrase_verb:2+fce-test) => (pronom_relatif:-that-which-whose)
        " " (phrase_verb:-copula+fqt+fsj-fin-fda)
        " " (preposition:+to)
* for +fpa, -fpa, +fpp, -fpp
* the vp in the right may or may not be passive:
* (copula_vp or does_vp constructs)
SYN 97 inverted_prep_proc

RULE terminal preposition with inverted RP prep_noun    (excluding to_noun)
(phrase_verb:2+fce-test) => (pronom_relatif:-that-which-whose)
        " " (phrase_verb:-copula+fqt+fsj-fin)
        " " (preposition:-to)
* for +fpa, -fpa, +fpp, -fpp
* the vp in the right may or may not be passive.
* (copula_vp or does_vp constructs)
SYN 97 inverted_prep_proc


***  Copula verb as an auxiliary modifier of the verb

RULE "is" as a modifier of a past participle forming the passive
(phrase_verb:i-copula-fcj+fph+fpa-fin) => (phrase_verb:+copula+is-fph) " "
        (phrase_verb:+fpp-fsj-fag-foj-fda-fpa-fvi-fce)
* e.g., "[Will the commander] be told [the reason?]",
*       "[coal] is carried [by the ship]"
SYN 11  copula_as_aux

RULE "have" as a modifier of an intransitive past participle
(phrase_verb:i-copula-fcj+fph-fpp-fin+fvi) => (phrase_verb:+have-fph) " "
        (phrase_verb:+fpp-fsj-fag-foj-fda-fpa+fvi)
* e.g., "[the ship] has sailed"
SYN 11  copula_as_aux

RULE "have" as a modifier of a transitive past participle
(phrase_verb:i-copula-fcj+fph-fpp-fin-fvi) => (phrase_verb:+have-fph) " "
        (phrase_verb:+fpp-fsj-fag-foj-fda-fpa-fvi)
* e.g., "[the commander] has told [the reason]"
SYN 11  copula_as_aux
```

```
**   Auxilary verb modification of the verb

RULE modifying the verb by "does" after question inversion
<phrase_verb:2-fcj+fph-fin+fqt+fsi> => <verbe_aux:+does>
        <phrase_verb:+foj-fqt+fsj>
* e.g., "does (the commander write reports)"
*       "est-ce que (la terre est ronde)"
*** added +foj to vp above Mar-27-84
SYN 3  no_op2_Verb_Common


**   Negation of the verb

RULE negating a verb
<phrase_verb:2-fcj+fph> => <adverbe:+not> " "
        <phrase_verb:-fsj-fag-foj-fdu-fvi>
* e.g., "[the commander did] not write [the report]"
*       "[John does] not have [the report]"
SYN 17  negative_proc

RULE negating a copula verb
<phrase_verb:1-fcj+fph> =>
        <phrase_verb:+copula-fph> " " <adverbe:+not>
* e.g., "[John] is not [an employee]"
SYN 17  negative_proc

RULE negating a copula verb
<phrase_verb:1-fcj+fph> =>
         "ne " <phrase_verb:+copula-fph> " " <adverbe:+not>
* e.g., "[John] is not [an employee]"
SYN 17  negative_proc

RULE negating a copula verb
<phrase_verb:1-fcj+fph> =>
         "n'" <phrase_verb:+copula-fph> " " <adverbe:+not>
* e.g., "[John] is not [an employee]"
SYN 17  negative_proc

RULE negating a copula verb after question inversion
<phrase_verb:1-fcj+fph> => <phrase_verb:+copula+fqt+fsj> " " <adverbe:+not>
* e.g., "(is John) not [an employee]"
SYN 17  negative_proc
```

```
**    "There" constructions

RULE "there" on a copula verb in question invertion position
<phrase_verb:+copula+is-fcj+fsj+fqt+fph+fag+fvi> =>
        <pronom:+il> " " <adverbe:+there> " a " <phrase_nom>
***     "Il y a [des vins roses]
* e.g., "is there [a female commander]"
SYN 2  is_there_proc

RULE "there" on a copula verb in question invertion position
<phrase_verb:+copula+is-fcj+fsj+fqt+fph+fag+fvi> =>
        <adverbe:+there> " a t'" <pronom:+il> " " <phrase_nom>
***     "Y a t'il [des vins roses]
* e.g., "is there [a female commander]"
SYN 2  is_there_proc

RULE "there" on a copula verb in question invertion position
<phrase_verb:+copula+is-fcj+fsj+fqt+fph+fag+fvi> =>
        <adverbe:+there> " a-t'" <pronom:+il> " " <phrase_nom>
***     "Y a-t'il [des vins roses]
* e.g., "is there [a female commander]"
SYN 2  is_there_proc

RULE wh-noun frontal there phrase
<phrase_verb:+is-copula-fcj+foj+fvi+fph+fsj> => <phrase_nom:-pof+qnf> " "
        <adverbe:+there> " a t'" <pronom:+il>
*** "(Quels vins) y a t'il"
* e.g., "(What ships) are there"
SYN 16 wh_is_there_proc

RULE wh-noun frontal there phrase
<phrase_verb:+is-copula-fcj+foj+fvi+fph+fsj> => <phrase_nom:-pof+qnf> " "
        <adverbe:+there> " a-t'" <pronom:+il>
*** "(Quels vins) y a-t'il"
* e.g., "(What ships) are there"
SYN 16 wh_is_there_proc

RULE wh-noun frontal there phrase with relative clause
<phrase_verb:+is-copula-fcj+foj+fvi+fph+fsj+fqt> => <phrase_nom:-pof+qnf> " "
        <adverbe:+there> " a t'" <pronom:+il> " "
        <prop_relative>
***     "(Quels vins) y a t'il (dont la couleur soit rouge)"
* e.g., "(What ships) are there (whose destination is Tokyo)"
*       "(What ships) are there (with Tokyo as destination)"
SYN 16 wh_is_there_proc
```

```
RULE wh-noun frontal there phrase with relative clause
<phrase_verb:+is-copula-fcj+foj+fvi+fph+fsj+fqt> => <phrase_nom:-pof+qnf> " "
        <adverbe:+there> " a-t'" <pronom:+il> " "
        <prop_relative>
k**     "(Quels vins) y a-t'il (dont la couleur soit rouge)"
* e.g., "(What ships) are there (whose destination is Tokyo)"
*       "(What ships) are there (with Tokyo as destination)"
SYN 16 wh_is_there_proc

(how <adj> constructions)

RULE how adj verb_phrase
<phrase_verb:3-copula+foj+fin-fcj-fpp> => <mot:+how> " " <adjectif:+num> " "
        <phrase_verb:+copula+fsj+fqt+fph+is>
* e.g., "how old [is John]"
SYN 2 how_adj_proc

RULE verb_phrase how adj
<phrase_verb:1-copula-fcj+foj+fph> => <phrase_verb:+copula+is-fsj-foj>
        " " <mot:+how> " " <adjectif:+num>
* e.g., "[John] is how old"
SYN 2 how_adj_proc

RULE adj_than_proc
<phrase_verb:1-copula-cjf+foj+fph> => <phrase_verb:+copula+is-fsj-foj>
        " " <adjectif:+comp> " " <mot:+than> " " <phrase_nom:-pof>
SYN 2 adj_than_proc
```

```
(ASK/FR/SNTAN.SYN  )

(  The Syntax of ASK French
   Sentence Rules for Questions   )


RULE sentence = transitive verb "?"
(sentence:+question) => (phrase_verb:+fsj+foj-fpp-fvi)
                (pct:+question_mark)
SYN 2  answer_proc

RULE sentence = intransitive verb "?"
(sentence:+question) => (phrase_verb:+fsj-fpp+fvi)
                (pct:+question_mark)
SYN 2  answer_proc

RULE "list" in form "what is"
(mot:+list+wh) => (mot_inter:-howmany) " "
                  (copule:+is)
SYN 3 no_opi_Sent_Answer

RULE list noun_phrase .
(sentence:+question) => (mot:+list-wh) " "
                (phrase_nom:-pof-pmf) "."
POST 4 fragment_answer_sem

RULE list noun_phrase ?
(sentence:+question) => (mot:+list-wh) " "
                (phrase_nom:-pof-pmf) (pct:+question_mark)
POST 4  fragment_answer_sem

RULE list noun_phrase .
(sentence:+question) => (mot:+list-wh) " "
                (phrase_nom:-pof+pmf-cjf) "."
POST 4 fragment_answer_sem

RULE list noun_phrase ?
(sentence:+question) => (mot:+list-wh) " "
                (phrase_nom:-pof+pmf-cjf) (pct:+question_mark)
POST 4  fragment_answer_sem

RULE list noun_phrase
(sentence:+question) (right_delimiter) => (mot:+list-wh) " "
          (phrase_nom:-pof) (right_delimiter)
POST 4  fragment_answer_sem

RULE sentence = "list" (vector) "."
(sentence:+question) => (mot:+list) " "
                (vecteur:-attribute) "."
SYN 2  answer_proc
```

```
RULE sentence = "list" (vector) "?"
(sentence:+question) => (mot:+list) " "
                        (vecteur:-attribute) (pct:+question_mark)
SYN 2  answer_proc

RULE sentence = "list" (vector)
(sentence:+question) (right_delimiter) => (mot:+list)
       " " (vecteur:-attribute) (right_delimiter)
SYN 2  answer_proc

RULE sentence = noun phrase "?"
(sentence:+question) => (phrase_nom:-unary_op) (pct:+question_mark)
SYN 2  answer_proc

RULE sentence = transitive verb (without "?")
(sentence:+question) (right_delimiter) =>
   (phrase_verb:+fsj+foj-fpp-fvi) (right_delimiter)
SYN 5   fragment_answer_proc

RULE sentence = intransitive verb (without "?")
(sentence:+question) (right_delimiter) =>
    (phrase_verb:+fsj-fpp+fvi) (right_delimiter)
POST 4  fragment_answer_sem

RULE sentence = noun phrase (without "?")
(left_delimiter) (sentence:+question) (right_delimiter) =>
    (left_delimiter) (phrase_nom:-unary_op) (right_delimiter)
POST 4  fragment_answer_sem
```

```
(ASK/FR/SNTIN.SYN  )

(  The Syntax of ASK French
   Sentence Rule for Data Input )


RULE input sentence = verb phrase "."
<sentence> => <phrase_verb:+fsj+foj-fpp-fvi> "."
PRE 2 input_proc

RULE remove from class
<sentence:+dec> => <mot:+remove> " " <phrase_nom>
   " " <preposition:+of> <nom:+class>
POST 2 remove_from_proc
```

```
(ASK/FR/SNTDE.SYN )

(     The Syntax of ASK French
      Sentence Rules for Definitions    )


RULE process definition - syntax part
<mot_nouv:+def> => <mot_nouv:+init+def>
SYN 3  def_proc_syn

RULE process definition - ":"
<mot_nouv:+def+colon> => <mot_nouv:-init+def-colon> ":"
SYN 2  no_op1_Sent_Definition

RULE process definition - ": "
<mot_nouv:+def+colon> => <mot_nouv:-init+def-colon> ": "
SYN 2  no_op1_Sent_Definition

RULE process definition - semantic part for nouns
<sentence> => <mot_nouv:+def+colon> <phrase_nom>
* e.g., "definition:N.Y.:New York"
*        "definition:f(x):3%x"
*        "definition:sscc:sin cos"
PRE 5  def_proc_sem

RULE process definition - semantic part for nouns
<sentence> => <mot_nouv:+def+colon> <unary_op>
PRE 5  def_proc_sem

RULE process definition - semantic part for vectors
<sentence> => <mot_nouv:+def+colon> <vecteur>
* e.g., "definition:opvec:cos, sin and tan"
*        "definition:attvec:length and width"
*        "definition:datvec:length and width of ships"
PRE 5  def_proc_sem
```

```
(ASK/FR/SNTME.SYN 1

  (   The Syntax of ASK French
      Sentence Rules to Find Out About the Data Base                    )

RULE comment
* Interprets an initial "***" as a signal for a comment line
<sentence> => "***"
SYN 2  comment_proc

RULE class of objects/classes/att_and_rel/units/variables
<phrase_nom:+class+meta> => <mot_nouv:+id>
SYN 2  class_of_proc

RULE class of objects/classes/att_and_rel/units/variables
<phrase_nom:+class+meta> => <mot_nouv:+id> " "
   <preposition:+of> <phrase_nom:-pof>
SYN 2  class_of_proc

RULE class of objects/classes/att_and_rel/units/variables
<phrase_nom:+class+meta> => <phrase_nom:+pof>
     " " <mot_nouv:+id>
SYN 2  class_of_proc

RULE units of
<phrase_nom:+number+unit> => <mot_nouv:+unit>
           " " <preposition:+of>
           <phrase_nom:+number+attribute>
POST 10  units_of_proc

RULE what is known about ...?
<sentence> => <sait_on> " " <phrase_nom:-number>
              <pct:+question_mark>
POST 5  known_proc

RULE what is known about ... "."
<sentence> => <sait_on> " " <phrase_nom:-number> "."
POST 5  known_proc

RULE what is known about ...
<sentence> <right_delimiter> => <sait_on> " "
           <phrase_nom:-number> <right_delimiter>
POST 5  known_proc
```

```
(ASK/FR/NUMBO.SYN)
*                Binary Operator Rules

* add features to rules that disallow
*    certain constructions with unary operators
*    eg, don't allow sgn or postfix to be on
* removed lit feature from noun_phrases
*    since it is never set there

* precedence features for numbers:
*    comp, cjf, to, psf (on left), pmf (on right), asf, mdf, pbj, exf, fnf

* binary_op_syn carries over class and
*    attribute feature

RULE addition and subtraction
(phrase_nom:i+asf) =>
    (phrase_nom:+number-sgn-postfix-comp-cjf-pof-psf-dtf)
    (binary_op:+asf)
    (phrase_nom:+number-sgn-postfix-comp-cjf-pof-pmf-asf)
SYN 2  binary_op_syn

RULE multiplication and division
(phrase_nom:i+mdf) =>
    (phrase_nom:+number-sgn-postfix-comp-cjf-pof-psf-asf-dtf)
    (binary_op:+mdf)
    (phrase_nom:+number-sgn-postfix-comp-cjf-pof-pmf-asf-mdf)
SYN 2  binary_op_syn

RULE exponentiation
(phrase_nom:i+exf) =>
    (phrase_nom:+number-sgn-postfix-comp-cjf-pof-psf-asf-mdf-fnf-pbj-dtf-exf)
    (binary_op:+exf)
    (phrase_nom:+number+attribute-sgn-postfix-comp-cjf-pof-pmf-asf-mdf-fnf-pbj)
* note: 2**2**3 should parse as "2 to the power 2 cubed", ie 2**(2**3)
SYN 2  binary_op_syn

RULE exponentiation
(phrase_nom:i+exf) =>
    (phrase_nom:+number-sgn-postfix-comp-cjf-pof-psf-asf-mdf-fnf-pbj-dtf-exf)
    (binary_op:+exf)
    (phrase_nom:+number-attribute-sgn-postfix-comp-cjf-pof-pmf-asf-mdf-pbj)
* note: sin**(cos(1))
SYN 2  binary_op_syn

RULE tack units onto a number: 3 meter, kgm meter
(phrase_nom:i+unit+pbj) =>
    (phrase_nom:+number-attribute-comp-psf-pmf-asf-mdf-dtf)" "
(phrase_nom:+number-attribute+unit-comp-cjf-psf-pmf-asf-mdf-pbj-dtf-class)
* this should allow "3,4 and 5 meters" to parse "(3,4 and 5) meters"
POST 2  binary_op_sem
```

```
(ASK/FR/NUMUO.SYN )

(Lexical rules)

* For the following two RULES, see Num_Misc

* RULE unary +
* <prim_unary_op:+sgn> => "+"
* PRE 2 op_1 (in ASK/UL/NUM.MI)

* RULE unary -
* <prim_unary_op:+sgn> => "-"
* PRE 2 op_1 (in ASK/UL/NUM.MI)

RULE make postfix uop
<prim_unary_op:i+postfix-cbl> => "au" <prim_unary_op:-postfix>
SYN 2 post_proc

RULE unary plus
<prim_unary_op:+sgn> => "plus"
LEX 1

RULE unary minus
<prim_unary_op:+sgn> => "moins"
LEX 2

RULE sin
<prim_unary_op> => "sin"
LEX 3

RULE sine
<prim_unary_op> => "sinus"
LEX 3

RULE cos
<prim_unary_op> => "cos"
LEX 4

RULE cosine
<prim_unary_op> => "cosinus"
LEX 4

RULE tan
<prim_unary_op> => "tan"
LEX 5

RULE tangent
<prim_unary_op> => "tangente"
LEX 5
```

```
RULE cotan
<prim_unary_op> => "cotan"
LEX 6

RULE arcsin
<prim_unary_op> => "arcsin"
LEX 7

RULE arccos
<prim_unary_op> => "arccos"
LEX 8

RULE arctan
<prim_unary_op> => "arctan"
LEX 9

RULE exp
<prim_unary_op> => "exp"
LEX 10

RULE exponential value
<prim_unary_op> => "exponentielle"
LEX 10

RULE ln
<prim_unary_op> => "ln"
LEX 11

RULE natural log
<prim_unary_op> => "log neperien"
LEX 11

RULE log to the base 10
<prim_unary_op> => "log"
LEX 12

RULE log10
<prim_unary_op> => "log10"
LEX 12

RULE sinh
<prim_unary_op> => "sinh"
LEX 13

RULE hyperbolic sine
<prim_unary_op> => "sinus hyperbolique"
LEX 13
```

```
RULE cosh
<prim_unary_op> => "cosh"
LEX 14

RULE hyperbolic cosine
<prim_unary_op> => "cosinus hyperbolique"
LEX 14

RULE tanh
<prim_unary_op> => "tanh"
LEX 15

RULE hyperbolic tangent
<prim_unary_op> => "tangente hyperbolique"
LEX 15

RULE abs
<prim_unary_op> => "abs"
LEX 20

RULE absolute value
<prim_unary_op> => "valeur absolue"
LEX 20

RULE trunc
<prim_unary_op> => "tronque"
LEX 21

RULE int
<prim_unary_op> => "ent"
LEX 21

RULE integer part
<prim_unary_op> => "partie entiere"
LEX 21

RULE integral part
<prim_unary_op> => "partie integrale"
LEX 21

RULE frac
<prim_unary_op> => "frac"
LEX 22

RULE fractional part
<prim_unary_op> => "partie fractionelle"
LEX 22
```

```
RULE round
<prim_unary_op> => "round"
LEX 23

RULE arr
<prim_unary_op> => "arr"
LEX 23

RULE valeur arrondie
<prim_unary_op> => "valeur arrondie"
LEX 23

RULE rac
<prim_unary_op> => "rac"
LEX 24

RULE rac
<prim_unary_op> => "racine"
LEX 24

RULE sqrt
<prim_unary_op> => "sqrt"
LEX 24

RULE square root
<prim_unary_op> => "racine carree"
LEX 24

RULE le sqr(x) = x*x
<prim_unary_op> -> "sqr"
LEX 25

RULE square
<prim_unary_op> => "carre"
LEX 25

RULE cube()
<prim_unary_op> => "cube"
LEX 26

;* Factor of 10 prefixes

RULE giga = billion
<prim_unary_op> => "giga"
LEX 27

RULE mega = million
<prim_unary_op> => "m*ga"
LEX 27
```

```
RULE kilo - thousand
<prim_unary_op> => "kilo"
LEX 37

RULE hecto - hundred
<prim_unary_op> => "hecto"
LEX 33

RULE deca - ten
<prim_unary_op> => "deca"
LEX 34

RULE deci - tenth
<prim_unary_op> => "deci"
LEX 35

RULE centi - hundredth
<prim_unary_op> => "centi"
LEX 36

RULE milli - thousandth
<prim_unary_op> => "milli"
LEX 37

RULE micro - millionth
<prim_unary_op> => "micro"
LEX 38

RULE nano - billionth
<prim_unary_op> => "nano"
LEX 39

RULE pico - *1.0e-12
<prim_unary_op> => "pico"
LEX 40

* statistical operators

* classin feature means that the operator
*   requires a class as input

RULE sum
<prim_unary_op:+classin> => "somme"
LEX 53

RULE total
<prim_unary_op:+classin> => "total"
LEX 54
```

```
RULE sum of squares
<prim_unary_op:+classin> => "somme des carres"
LEX 52

RULE mean
<prim_unary_op:+classin> => "moyenne"
LEX 53

RULE variance
<prim_unary_op:+classin> => "variance"
LEX 54

RULE standard deviation
<prim_unary_op:+classin> => "ecart type"
LEX 55

RULE sigma
<prim_unary_op:+classin> => "sigma"
LEX 56

RULE standard error
<prim_unary_op:+classin> => "erreur moyenne"
LEX 56

RULE std err
<prim_unary_op:+classin> => "err moy"
LEX 56

RULE maximum
<prim_unary_op:+classin> => "maximum"
LEX 57

RULE max
<prim_unary_op:+classin> => "max"
LEX 57

RULE minimum
<prim_unary_op:+classin> => "minimum"
LEX 58

RULE min
<prim_unary_op:+classin> => "min"
LEX 58

RULE range
<prim_unary_op:+classin> => "domaine"
LEX 59
```

```
RULE skewness
<prim_unary_op:+classin> => "moment d'ordre 3"
LEX 60

RULE kurtosis
<prim_unary_op:+classin> => "moment d'ordre 4"
LEX 61

(remove spaces around unary operators)

RULE space - unary op
<prim_unary_op:1+cbl> => " " <prim_unary_op>
SYN 2 no_op1_Num_Unary_Oper

RULE unary op - space
<prim_unary_op:1> => <prim_unary_op:-cbl> " "
SYN 2 no_op1_Num_Unary_Oper

(Syntax for Arithmetic Unary Operator Rule     )

* unary_op_syn will carry over all features
*    of the prim_unary_op
* (postfix, classin, syn )

RULE develop unary operator
<phrase_nom:+number+attribute+unary_op> =>
          <prim_unary_op>
SYN 8   unary_op_syn
```

```
[ASK/FR/NUMMI.SYN]

          (    Miscellaneous Number Rules    )

RULE noun_phrase:+number => number
(phrase_nom:+number+entity-lit) => (number:-conj-lit)
SYN 2 no_op1_Num_Misc

RULE noun_phrase:+number => number
(phrase_nom:+number+entity+lit) => (number:-conj+lit)
SYN 2 no_op1_Num_Misc

RULE 1.0e3
(phrase_nom:+number+entity) => (number)"e"(whole_number)
SYN 2 exp_format_plus

RULE 1.0e+3
(phrase_nom:+number+entity) => (number)"e+"(whole_number)
SYN 2 exp_format_plus

RULE 1.0e-3
(phrase_nom:+number+entity) => (number)"e-"(whole_number)
SYN 2 exp_format_minus

RULE 1.0E3
(phrase_nom:+number+entity) => (number)"E"(whole_number)
SYN 2 exp_format_plus

RULE 1.0E+3
(phrase_nom:+number+entity) => (number)"E+"(whole_number)
SYN 2 exp_format_plus

RULE 1.0E-3
(phrase_nom:+number+entity) => (number)"E-"(whole_number)
SYN 2 exp_format_minus

RULE unary +
(prim_unary_op:+sgn) => "+"
PRE 2 op_1

RULE unary -
(prim_unary_op:+sgn) => "-"
PRE 2 op_1

RULE binary +
(binary_op:+asf) => "+"
PRE 2 op_1
```

```
RULE binary -
<binary_op:+asf> => "-"
PRE 2 op_1

RULE multiplication: *
<binary_op:+mdf+mult> => "*"
PRE 2 op_1

RULE division: /
<binary_op:+mdf> => "/"
PRE 2 op_1

RULE exponentiation: **
<binary_op:+exf> => "**"
PRE 2 op_1

RULE exponentiation: ^
<binary_op:+exf> => "^"
PRE 2 op_1

RULE remove left space from binary op
<binary_op:i+cbl> => " "<binary_op>
POST 2 no_op1_Num_Misc

RULE remove right space from binary op
<binary_op:i> => <binary_op:-cbl>" "
POST 2 no_op1_Num_Misc

RULE "=", no. 0
<comparator:+is+sym> => "="
PRE 2 op_1

RULE ">", no. 1
<comparator:+is+sym> => ">"
PRE 2 op_1

RULE "<", no. 2
<comparator:+is+sym> => "<"
PRE 2 op_1

RULE ">=", no. 3
<comparator:+is+sym> => ">="
PRE 2 op_1

RULE "<=", no. 4
<comparator:+is+sym> => "<="
PRE 2 op_1
```

```
RULE "<>", no. 5
<comparator:+is+sym> => "<>"
PRE 2 op_1

RULE remove left space from comparator symbol
<comparator:1+cbl> => " " <comparator:+sym>
SYN 2 no_op1_Num_Misc

RULE remove right space from comparator symbol
<comparator:1> => <comparator:+sym-cbl> " "
SYN 2 no_op1_Num_Misc


**   Rules that put a comparator onto a number or number attribute
**   don't worry yet about classes causing trouble

* comparators +is (can take the place of is) are handled in verb_phrase
* rules (this includes all comparator symbols)
* these rules thus get ready for "<num> is <num>?" rule

RULE add comparator to number
<phrase_nom:2+comp-variable> => <comparator:-between-than-is-sym-cbl>
       " " <phrase_nom:+number-comp>
POST 2 comp_num

RULE between num and num
<phrase_nom:2+comp-variable> => <comparator:+between-cbl> " "
       <phrase_nom:+number-comp> " " "
       <phrase_nom:+number-comp>
POST 2 comp_num

RULE add comparator to number
<phrase_nom:3+comp-variable> => <comparator:+than-cbl> " "
       <not:+than> <phrase_nom:+number-comp-attribute>
POST 2 comp_num


*** RULEs to build number conjunctions by

RULE num to num
<phrase_nom:+number+class> =>
       <phrase_nom:+number-class-attribute-comp-dtf>
       " " <preposition:+to>
       <phrase_nom:+number-class-attribute-comp>
POST 2 num_to_num
```

```
RULE num to num by num
(phrase_nom:+number+class) =>
   (phrase_nom:+number-class-attribute-comp-dtf)
   " " (preposition:+to)
   (phrase_nom:+number-class-attribute-comp)
   " " (preposition:+by)
   (phrase_nom:+number-class-attribute-comp)
POST 2 num_to_num

RULE for
(phrase_nom:+number+class) =>
(phrase_nom:+number-class-comp-dtf) " " (preposition:+for)
(phrase_nom:+number+variable) "=" (phrase_nom:+number-attribute-comp)
SYN 2  num_for_num_syn


*** Unit rule

RULE units are numbers
(phrase_nom:+number+unit) => (nom:+unit)
POST 2 unit_to_num


*** Literal number conversions

RULE lit num "s"
(literal_number:i) => (literal_number) "s"
SYN 2 no_opi_Num_Misc

RULE lit to num: +dig1
(number:+lit+dig1-conj) => (literal_number:+dig1-conj)
SYN 2 lit_to_num

RULE lit to num: +conj
(number:+lit+dig1+conj) => (literal_number:+dig1+conj)
SYN 2 lit_to_num

RULE lit to num: +nbs
(number:+lit+nbs) => (literal_number:-dig2+nbs)
SYN 2 lit_to_num

RULE lit to num: +dig2
(number:+lit+dig2-nbs) => (literal_number:+dig2-nbs)
SYN 2 lit_to_num
```

```
RULE lit to num: +dig2+nbs
(number:+lit+dig2+nbs) => (literal_number:+dig2+nbs)
SYN 2 lit_to_num

RULE lit to num: +big
(number:+lit+big) => (literal_number:+big)
SYN 2 lit_to_num

RULE lit to num
(number:+lit) => (literal_number:-big-dig1-dig2-nbs)
SYN 2 lit_to_num

* combine literal numbers

RULE one million, one thousand, one hundred
(number:+lit+big+lit2) => (number:+lit-big) " " (number:+lit+big-lit3-lit2-lit1)
SYN 2  num_multi

RULE one hundred twenty three
(number:+lit-big+lit1) => (number:+lit+big+lit2) " "
                          (number:+lit-lit1)
SYN 2 num_add

RULE one thousand one hundred
(number:+lit+big+lit2+lit3) => (number:+lit+big+lit2-lit3) " "
                               (number:+lit+big+lit2)
SYN 2 num_add

RULE hundred thousand
(number:+lit+big+lit3) => (number:+lit+big-lit2-lit3) " " (number:+lit+big-lit2)
SYN 2  num_multi

RULE thousand hundred
(number:+lit+big+lit3) => (number:+lit+big-lit2-lit3) " " (number:+lit)
SYN 2  num_add

RULE thirty six
(number:+lit) => (number:+lit+dig2)" "(number:+lit+dig1)
SYN 2  num_add

RULE soixante treize
(number:+lit) => (number:+lit+dig2+nbs)" "(number:+lit+nbs)
SYN 2  num_add
```

```
** literal to whole number conversions, as above for reals

RULE literal to whole number: +dig1
<whole_number:+dig1-art> => <literal_number:-real+dig1-art>
SYN 2 no_opt_Num_Misc

RULE literal to whole number: +dig1
<whole_number:+dig1+art> => <literal_number:-real<dig1+art>
SYN 2 no_opt_Num_Misc

RULE literal to whole number: +dig2
<whole_number:+dig2> => <literal_number:-real+dig2>
SYN 2 no_opt_Num_Misc

RULE literal to whole number: +big
<whole_number:+big> => <literal_number:-real+big>
SYN 2 no_opt_Num_Misc

RULE literal to whole number
<whole_number> => <literal_number:-real-dig1-dig2-big>
SYN 2 no_opt_Num_Misc

* combine literal whole numbers

RULE one million, one thousand, one hundred
<whole_number:+big+lit2> => <whole_number:-big> " "
                            <whole_number:+big-lit2-lit1>
SYN 2 wh_num_mult

RULE one hundred twenty three
<whole_number:-big+lit1> => <whole_number:+big+lit2> " "
                            <whole_number:-big-lit1>
SYN 2 wh_num_add

RULE one thousand one hundred
<whole_number:+big+lit2+lit3> => <whole_number:+big+lit2-lit3> " "
                                 <whole_number:+big+lit2>
SYN 2 wh_num_add

RULE hundred thousand
<whole_number:+big+lit3> => <whole_number:+big-lit2-lit3> " "
                            <whole_number:+big-lit2>
SYN 2 wh_num_mult

RULE thirty six
<whole_number> => <whole_number:+dig2>" "<whole_number:+dig1>
SYN 2 wh_num_add
```

```
** ordinal numbers

RULE add wh_num to ordinal
<ordinal> => <whole_number>" "<ordinal>
POST 2 wh_num_add

RULE whole number to ordinal
<ordinal> => <whole_number> "er"
SYN 2  no_op1_Num_Misc

RULE whole number to ordinal
<ordinal> => <whole_number> "eme"
SYN 2  no_op1_Num_Misc

RULE whole number to ordinal
<ordinal> => <whole_number> "ieme"
SYN 2  no_op1_Num_Misc

RULE whole number to ordinal
<ordinal> => <whole_number> "nd"
SYN 2  no_op1_Num_Misc

RULE sentence.
<sentence> => <sentence>"."
POST 2 default
```

```
(ASK/SL/COM.SYN)

(Rules in the COMMAND and META languages are created in bilingual pairs
so as to avoid the generation of two distinct dictionaries for so small
vocabularies)


RULE exit
<sentence> => <exit>
PRE 3 exit_proc

RULE exit
<exit> => "sortez"
LEX 0

RULE exit
<exit> => "sors"
LEX 0

RULE exit
<exit> => "exit"
LEX 0

RULE exit to identifier
<sentence> => "exit to " <identifier>
PRE 4 exit_to_proc

RULE exit to identifier
<sentence> => "sort en " <identifier>
PRE 4 exit_to_proc

RULE exit to identifier
<sentence> => "sortez en " <identifier>
PRE 4 exit_to_proc

RULE enter identifier
<sentence> => "enter " <identifier>
PRE 5 enter_proc

RULE enter identifier
<sentence> => "entrez en " <identifier>
PRE 5 enter_proc

RULE enter identifier
<sentence> => "entre en " <identifier>
PRE 5 enter_proc

RULE $ENT$ identifier
<sentence> => "$ENT$" <identifier>
PRE 5 enter_proc
```

```
RULE $ENT$ sentence
<sentence> => "$ENT$" <sentence>
POST 2 default

RULE enter
<enter> => "enter"
LEX 0

RULE enter
<enter> => "entre"
LEX 0

RULE enter
<enter> => "entrez"
LEX 0

RULE enter
<enter> => "entrer en"
LEX 0

RULE enter
<sentence> => <enter>
PRE 6 enter_what_proc

RULE enter metalanguage identifier
<sentence> => "enter meta " <identifier>
PRE 7 enter_meta_proc

RULE enter metalanguage identifier
<sentence> => "enter META " <identifier>
PRE 7 enter_meta_proc

RULE enter metalanguage identifier
<sentence> => "entre en meta " <identifier>
PRE 7 enter_meta_proc

RULE enter metalanguage identifier
<sentence> => "entrez en meta " <identifier>
PRE 7 enter_meta_proc

RULE enter metalanguage identifier
<sentence> => "entre en META " <identifier>
PRE 7 enter_meta_proc

RULE enter metalanguage identifier
<sentence> => "entrez en META " <identifier>
PRE 7 enter_meta_proc
```

```
RULE end ASK
<sentence> => "end ASK"
PRE 9 endPOL_proc

RULE end ASK
<sentence> => "arrete ASK"
PRE 9 endPOL_proc

RULE end ASK
<sentence> => "arretez ASK"
PRE 9 endPOL_proc

RULE logon sentence
<sentence> => <auth user>
POST 10 login_proc

RULE change password
<chpswd> => "change password"
LEX 0

RULE change password
<chpswd> => "changez mot de passe"
LEX 0

RULE change password
<chpswd> => "change le mot de passe"
LEX 0

RULE change password
<chpswd> => "changez le mot de passe"
LEX 0

RULE change password
<chpswd> => "change mot de passe"
LEX 0

RULE chpswd
<sentence> => <chpswd>
PRE 11 change_password

RULE authorize
<authorize> => "authorize"
LEX 0

RULE authorize
<authorize> => "autorise"
LEX 0
```

```
RULE autorise
<authorize> => "autorisez"
LEX 0

RULE auth id
<sentence> => <authorize> " " <identifier>
PRE 12 authorize_proc

RULE deauthorize
<deauth> => "deauthorize"
LEX 0

RULE deauthorize
<deauth> => "desautorise"
LEX 0

RULE deauthorize
<deauth> => "desautorisez"
LEX 0

RULE deauth id
<sentence> => <deauth> " " <auth_user:-mult>
PRE 13 deauthorize_proc

RULE no one
<auth_user:+mult> => "no one"
LEX 32

RULE anyone
<auth_user:+mult> => "anyone"
LEX 33

RULE no one
<auth_user:+mult> => "personne"
LEX 32

RULE anyone
<auth_user:+mult> => "n'importe qui"
LEX 33

RULE enter auth
<sentence:+ent> => <auth_user> " is " <authorize> "d "
                   <word:+to> " " <enter> " " <identifier>
POST 14 auth_entry_proc
```

```
RULE enter auth
<sentence:+ent> => <authorize> " " <auth_user> " "
                   <word:+to> " " <enter> " " <identifier>
POST 14 auth_entry_proc

RULE enter auth
<sentence:+ent+not> => <auth_user:-mult> " is not " <authorize> "d "
                   <word:+to> " " <enter> " " <identifier>
POST 14 auth_entry_proc

RULE base auth
<sentence:+base> => <auth_user> " is " <authorize> "d "
                   <word:+to> " " <base> " " <word:+on> " " <identifier>
POST 14 auth_entry_proc

RULE base auth
<sentence:+base> => <authorize> " " <auth_user> " "
                   <word:+to> " " <base> " " <word:+on> " " <identifier>
POST 14 auth_entry_proc

RULE base auth
<sentence:+base+not> => <auth_user:-mult> " is not " <authorize> "d "
                   <word:+to> " " <base> " " <word:+on> " " <identifier>
POST 14 auth_entry_proc

RULE enter auth
<sentence:+ent> => <auth_user> " est " <authorize> " "
                   <word:+to> " " <enter> " " <identifier>
POST 14 auth_entry_proc

RULE enter auth
<sentence:+ent> => <authorize> " " <auth_user> " "
                   <word:+to> " " <enter> " " <identifier>
POST 14 auth_entry_proc

RULE enter auth
<sentence:+ent+not> => <auth_user:-mult> " n'est pas " <authorize> " "
                   <word:+to> " " <enter> " " <identifier>
POST 14 auth_entry_proc

RULE base auth
<sentence:+base> => <authorize> " " <auth_user> " "
                   <word:+to> " " <base> " " <word:+on> " " <identifier>
POST 14 auth_entry_proc
```

```
RULE base auth
<sentence:+base> => <auth_user> " est " <authorize> " "
                    <word:+to> " " <base> " " <word:+on> " " <identifier>
POST 14 auth_entry_proc

RULE base auth
<sentence:+base+not> => <auth_user:-mult> " n'est pas " <authorize> " "
                    <word:+to> " " <base> " " <word:+on> " " <identifier>
POST 14 auth_entry_proc

RULE delete identifier
<sentence> => "delete " <identifier>
PRE 15 delete_proc

RULE delete identifier
<sentence> => "supprime " <identifier>
PRE 15 delete_proc

RULE delete identifier
<sentence> => "supprimez " <identifier>
PRE 15 delete_proc

RULE lex directory
<directory> => "directory"
LEX 0

RULE lex directory
<directory> => "repertoire"
LEX 0

RULE directory
<sentence> => <directory>
PRE 16 directory_proc

RULE extend
<extend> => "extend"
LEX 0

RULE ex
<extend> => "ex"
LEX 0

RULE extend
<extend> => "prolonge"
LEX 0
```

```
RULE ex
<extend> => "prolongez"
LEX 0

RULE extend short
<sentence> => <extend> " " <identifier> " "
              <file_spec:-vol> ":" <identifier>
POST 17 extend_proc

RULE extend short
<sentence> => <extend> " " <identifier> " "
              <file_spec:-vol> ":" <file_spec>
POST 17 extend_proc

RULE extend short
<sentence> => <extend> " " <identifier> " "
              <identifier> ":" <identifier>
POST 17 extend_proc

RULE extend short
<sentence> => <extend> " " <identifier> " "
              <identifier> ":" <file_spec>
POST 17 extend_proc

RULE with
<with> => "with"
LEX 0

RULE with
<with> => "avec"
LEX 0

RULE sentence extend
<sentence> => <extend> " " <identifier> " " <with> " " <file_spec:-vol>
              ":" <identifier>
POST 17 extend_proc

RULE sentence extend
<sentence> => <extend> " " <identifier> " " <with> " " <file_spec:-vol>
              ":" <file_spec>
POST 17 extend_proc

RULE sentence extend
<sentence> => <extend> " " <identifier> " " <with> " " <identifier>
              ":" <identifier>
POST 17 extend_proc
```

```
RULE sentence extend
<sentence> => <extend> " " <identifier> " " <with> " " <identifier>
              ":" <file_spec>
POST 17 extend_proc

RULE base
<base> => "base"
LEX 0

RULE base
<base> => "basez"
LEX 0

RULE base
<base> => "baser"
LEX 0

RULE BASE ON
<sentence> => <base> " " <identifier> " "
                     <word:+on> " " <identifier>
PRE 18 base_proc

RULE unbase
<unbase> => "unbase"
LEX 0

RULE unbase
<unbase> => "debase"
LEX 0

RULE unbase
<unbase> => "debasez"
LEX 0

RULE unbasing
<sentence> => <unbase> " " <identifier> " " <word:+from> " " <identifier>
PRE 19 unbase_proc

RULE hardcopy
<word:+hardcopy> => "hardcopy"
LEX 0

RULE hardcopy
<word:+hardcopy> => "imprimante"
LEX 0
```

```
RULE hardcopy
<word:+hardcopy> => "l'imprimante"
LEX 0

RULE printer is
<word:+hardcopy> => "printer is"
LEX 0

RULE printer is
<word:+hardcopy> => "l'imprimante est"
LEX 0

RULE on
<word:+on+onoff> => "on"
LEX 1

RULE off
<word:+onoff> => "off"
LEX 1

RULE on
<word:+on+onoff> => "sur"
LEX 1

RULE on
<word:+on+onoff> => "branche"
LEX 1

RULE off
<word:+onoff> => "debranche"
LEX 0

RULE hardcopy on/off
<sentence> => <word:+hardcopy> " " <word:+onoff>
PRE 20 hardcopy

RULE hardcopy on/off
<sentence> => <word:+onoff> " " <word:+hardcopy>
PRE 20 branche_imprimante

RULE hardcopy on/off
<sentence> => <word:+onoff> "z " <word:+hardcopy>
PRE 20 branche_imprimante

RULE hardcopy on/off
<sentence> => <word:+hardcopy> " " <word:+onoff> "e"
PRE 20 hardcopy
```

```
RULE file spec =) .<id>
 <file_spec:-vol-unix+dot>=) "." <identifier>
POST 21 file_spec_const_proc

RULE file spec =) <id>.<No>
(file_spec:-vol-unix-dot) =) <identifier> "."  <whole_number>
POST 21 file_spec_const_proc

RULE file spec =) <id>.<id>
<file_spec:-vol-unix-dot> =) <identifier> <file_spec:-vol-unix+dot>
POST 21 file_spec_const_proc

RULE file spec =) <file_spec>.<id>
<file_spec:1) =) <file_spec:-vol-unix) <file_spec:-vol-unix+dot>
POST 21 file_spec_const_proc

RULE file spec =) "/"<id>
<file_spec:-vol+slash+unix) <right_delimiter>
        =) "/" <identifier) <right_delimiter>
POST 21 file_spec_const_proc

RULE file spec =) "/"<file_spec>
<file_spec:-vol+slash+unix) =) "/" <file_spec:-vol>
POST 21 file_spec_const_proc

RULE file spec =) <id>"/"<id>
<file_spec:-vol-slash+unix) =) <identifier) <file_spec:-vol+slash>
POST 21 file_spec_const_proc

RULE file spec =) <fs>"/"<fs>
<file_spec:-vol-slash+unix) =>
  <file_spec:-vol-unix-slash) <file_spec:-vol+slash>
POST 21 file_spec_const_proc

RULE file spec =) <id>"_"<id>
<file_spec:-vol-slash-unix) =) <identifier) "_" <identifier>
POST 21 file_spec_const_proc

RULE file spec =) <file_spec>"_"<id>
<file_spec:1) =) <file_spec:-vol-unix) "_" <identifier>
POST 21 file_spec_const_proc

RULE file spec:+vol
<file_spec:+vol) =) <identifier) ":" <file_spec:-vol>
POST 21 file_spec_const_proc
```

```
RULE file spec:+vol
<file_spec:+vol> => <file_spec:-vol> ":" <file_spec:-vol>
POST 21 file_spec_const_proc

RULE file spec:+vol (number)
<file_spec:+vol> => "#" <whole_number> ":" <file_spec:-vol>
POST 21 file_spec_const_proc

RULE file spec:+vol
<file_spec:+vol> => <identifier> ":" <identifier>
POST 21 file_spec_const_proc

RULE file spec:+vol
<file_spec:+vol> => <file_spec:-vol> ":" <identifier>
POST 21 file_spec_const_proc

RULE file spec:+vol (number)
<file_spec:+vol> => "#" <whole_number> ":" <identifier>
POST 21 file_spec_const_proc

RULE read
<altin:+read> => "read"
LEX 0

RULE read
<altin:+read> => "lis"
LEX 0

RULE read
<altin:+read> => "lisez"
LEX 0

RULE from
<word:+from> => "from"
LEX 0

RULE from
<word:+from> => "le"
LEX 0

RULE from
<word:+from> => "de"
LEX 0

RULE file
<word:+file> => "file"
LEX 0
```

```
RULE file
<word:+file> => "fichier"
LEX 0

RULE read from file => ""
<altin:+comp> => <altin:+read> " " <word:+from>
 " " <word:+file>
SYN 2  default

RULE read from => ""
<altin:+comp> => <altin:+read> " " <word:+from>
SYN 2  default

RULE read from => ""
<altin:+comp> => <altin:+read> " "
SYN 2  default

RULE alternate input lexical rule
<altin:+comp> => "alternate input is"
LEX 0

RULE alternate input lexical rule
<altin:+comp> => "entree alterne est"
LEX 0

RULE read from file file_spec
<sentence> => <altin:+comp> " " <file_spec>
POST 22  read_from_proc

RULE read from file file_spec
<sentence> => <altin:+comp> " " <identifier>
POST 22  read_from_proc

RULE close
<close> => "close output file"
LEX 0

RULE close
<close> => "close alternate output"
LEX 0

RULE close
<close> => "ferme le fichier de sortie"
LEX 0

RULE close
<close> => "ferme la sortie alterne"
LEX 0
```

```
RULE close
<close> => "fermez le fichier de sortie"
LEX 0

RULE close
<close> => "fermez la sortie alterne"
LEX 0

RULE stop file output
<sentence> => <close>
PRE 23 close alt_out

RULE write
<altout:+write> => "write"
LEX 0

RULE write
<altout:+write> => "ecris"
LEX 0

RULE write
<altout:+write> => "ecrivez"
LEX 0

RULE to
<word:+to+thru> => "to"
LEX 0

RULE to
<word:+to+thru> => "a"
LEX 0

RULE alt_out -> write to file
<altout:+comp> => <altout:+write> " " <word:+to> " " <word:+file> " "
SYN 2 default

RULE alternate output is
<altout:+comp> => "output file is"
LEX 0

RULE alternate outout is
<altout:+comp> => "alternate output is"
LEX 0

RULE alternate outout is
<altout:+comp> => "fichier de sortie est"
LEX 0
```

```
RULE alternate output is
<altout:+comp> => "sortie alterne est"
LEX 0

RULE output sem
<sentence> => <altout:+comp> " " <identifier>
POST 24 open_file_output

RULE output sem
<sentence> => <altout:+comp> " " <file_spec>
POST 24 open_file_output

RULE through
<word:+thru> => "through"
LEX 0

RULE thru
<word:+thru> => "thru"
LEX 0

RULE through
<word:+thru> => "jusqu'a"
LEX 0

RULE $OUT$
<pref:+out> => "$OUT$"
SYN 2 no_op0

RULE $AMB$
<pref:+amb> => "$AMB$"
SYN 2 no_op0

RULE $OUT$(sp), $AMB$(sp)
<pref:1> => <pref> " "
SYN 2 no_op0

RULE print line range
<sentence> => <pref:+out> <whole_number> " " <word:+thru> " " <whole_number>
PRE 26 line_range

RULE print line range
<sentence> => <pref:+out> "de "  <whole_number>
          " " <word:+thru>  " "  <whole_number>
PRE 26 line_range

RULE print line range " - "
<sentence> => <pref:+out> <whole_number> " - " <whole_number>
PRE 26 line_range
```

```
RULE print line range "-"
<sentence> => <pref:+out> <whole_number> "-" <whole_number>
PRE 26 line_range

RULE print next n lines
<sentence> <right_delimiter> => <pref:+out>
                  <whole_number> <right_delimiter>
PRE 27 next_n_lines

RULE all
<word:+all> => "all"
LEX 0

RULE all
<word:+all> => "tous"
LEX 0

RULE all
<word:+all> => "tout"
LEX 0

RULE all
<word:+all> => "toute"
LEX 0

RULE all
<word:+all> => "toutes"
LEX 0

RULE print all lines
<sentence> => <pref:+out> <word:+all>
PRE 2 all_lines

RULE rest
<word:+rest> => "rest"
LEX 0

RULE rest
<word:+rest> => "reste"
LEX 0

RULE rest
<word:+rest> => "le reste"
LEX 0

RULE print rest of lines
<sentence> => <pref:+out> <word:+rest>
PRE 2 rest_of_lines
```

```
RULE none
<word:+none> => "none"
LEX 0

RULE none
<word:+none> => "aucun"
LEX 0

RULE none
<word:+none> => "aucuns"
LEX 0

RULE none
<word:+none> => "aucune"
LEX 0

RULE none
<word:+none> => "aucunes"
LEX 0

RULE print no lines
<sentence> => <pref:+out> <word:+none>
PRE 2 no_lines

RULE print ambiguity
<sentence> => <pref:+amb> <whole_number>
PRE 2 which_amb

RULE dont print ambiguity
<sentence> => <pref:+amb> <word:+none>
PRE 2 no_ambiguity

RULE print all ambiguities
<sentence> => <pref:+amb> <word:+all>
PRE 2 all_ambiguity

RULE ignore lines or ambiguities
<sentence> <right_delimiter> => <pref> <right_delimiter>
POST 2 out_default

RULE ignore lines or ambiguities, do sentence
<sentence> => <pref> <sentence>
POST 2 out_default

RULE help
<word:+help> => "help"
LEX 0
```

```
RULE help
<word:+help> => "aide"
LEX 0

RULE help sent
<sentence> => <word:+help>
POST 35 help_proc

RULE help on sent
<sentence> => <word:+help> " " <word:+topic>
POST 35 help_proc

RULE French
<language> => "French"
LEX 2

RULE French
<language> -> "Francais"
LEX 2

RULE English
<language> => "English"
LEX 1

RULE English
<language> => "Anglais"
LEX 1

RULE change
<word:+change> => "change"
LEX 0

RULE change
<word:+change> => "changez"
LEX 0

RULE change language
<left_delimiter> <sentence> <right_delimiter> =>
                          <left_delimiter> <language> <right_delimiter>
POST 36 chg_lang_proc

RULE change language
<sentence> => <word:+change> " language to " <language>
POST 37 chg_lang_proc

RULE change language
<sentence> => <word:+change> " language to " <language> "."
POST 38 chg_lang_proc
```

```
RULE change language
<sentence> => <word:+change> " la langue au " <language>
POST 37 chg_lang_proc

RULE change language
<sentence> => <word:+change> " la langue au " <language> ","
POST 38 chg_lang_proc

RULE change language
<sentence> => <word:+change> " la langue a l'" <language>
POST 37 chg_lang_proc

RULE change language
<sentence> => <word:+change> " la langue a l'" <language> ","
POST 38 chg_lang_proc
```